Comprehensive Overview: Application of Graph Theory

***Dr. Om Prakash Dave**

Abstract

Graph theory is use for numerous branches of science, technology, and mathematics is making it a rapidly expanding field. Computer science (algorithms and computing), chemistry, biochemistry, networks of communication, coding theory, and operations research (scheduling) are among the fields that actively employ it. In addition, graph theory is used in database administration, circuit design, coding theory, radar, x-ray crystallography, astronomy, and communication network addressing. The applications of graph theory in various diverse domains are summarized in this work, with an emphasis on computer science and chemistry applications that make use of graph theoretical ideas. This overview and summary cover a number of graph theory papers pertaining to scheduling, computer science applications, and chemistry.

Keywords: graph theory, applications, chemistry, isomerism, algorithms

Introduction

Graph theory is a study of graphs, which are mathematical frameworks used to depict pairwise interactions between things, in the fields of computer science, math, and chemistry. In this context, a "graph" is made up of "vertices" or "nodes" and the lines that connect them, known as edges. For more thorough definitions and other variations in the kinds of graphs that are commonly considered, see graph (mathematics). A graph can be undirected, which means that there does not exist difference between the two vertices associated with each edge, or it can have edges that are directed from one vertex to another. One of the main topics of study in discrete mathematics is graphs. Generally speaking, a graph is an ordered pair $G = (V, E)$ that consists of a set of vertices or nodes and a set of edges or lines that are 2-element subsets (that is, an edge is related to two vertices, and the connection is represented as an unorganized set of the vertices with respect to the particular edge).

The Königsberg Bridge problem in 1735 served as the catalyst for the development of graph theory. This issue gave rise to the idea of the Eulerian Graph. Euler researched the Königsberg Bridge issue and developed a framework, known as an Eulerian graph, to address it. Kuratowski used recreational problems to demonstrate the planarity of complete graphs and bipartite graphs, which were first proposed by A.F. Möbius in 1840. In 1845, Gustav Kirchhoff used graph theory concepts to calculate currents in electrical networks or circuits, using the idea of a tree—a connected graph devoid of cycles. Thomas Guthrie found the well-known four-color dilemma in 1852. Later, in 1856, Thomas P.

Kirkman and William R. Hamilton established the idea of a Hamiltonian graph by examining pathways that make exactly one visit to specific locations while investigating cycles on polyhedra. An important turning point in the history of graph theory was reached in 1913 when H. Dudeney described a puzzle problem using graphs. Despite the formulation of the four-color problem, it wasn't until a century later that Kenneth Appel and Wolfgang Haken solved it.

In order to study trees, Cayley looked at certain analytical forms from differential calculus. This led to the invention of enumerative graph theory and had several consequences in theoretical chemistry. In 1878, Sylvester coined the term "graph" after drawing a comparison between "quantic invariants" and covariants of algebra and molecular diagrams. Ramsey's work on colorations in 1941 gave rise to extremal graph theory, a subfield of graph theory. Heinrich used computers to solve the four-color problem in 1969. Random graph theory emerged from the investigation of asymptotic graph connectedness. Combinatorial and topological approaches in nonlinear chemical kinetics, for instance, have demonstrated that graph theory can assist in figuring out the dynamics of complex chemical reactions, like oscillatory reactions, and in graph theoretical models for potential mechanisms of a particular type of reaction. Combinatorial methods are also useful in chemical kinetics.

Graphs in Chemistry

Constitutional graphs are all structural formulations of covalently bonded substances, which are graphs. The lines (edges of the graph) in their constitutional formulas represent covalent two-electron bonds, and over 90% are organic or include organic ligands. There is only one kind of graph that chemists are interested in: constitutional graphs. The purpose of this analysis is to demonstrate how graphs are used in chemistry. Definition, enumeration, organization, codification, nomenclature as correlation, and programming for computers are all based on graph theory.

Structural formulas are linked to chemical information and can be uniquely and methodically indexed and retrieved. Nomenclature conventions are used to transform chemical structures into words, albeit words typically follow chemical information. The phenomena of isomerism, which is explained by chemical structure theory, is the main reason why graph theory (GT) is important for chemistry. This theory uses only graph-theoretical techniques to account for all constitutional isomers.

Constitutional isomer definition and identification (corresponding to a particular molecular formula) are strictly graph-theoretical processes. The main problem in the documentation and retrieval process is isomerism. This is where sophisticated graph theory techniques can be useful, even though chemical formulas can be arranged for indexing according to straightforward criteria.

The combinatorics of atoms, which are subject to certain principles, is the fundamental aspect of chemistry. Prior to recently, the majority of theoretical chemists thought of mathematics as an instrument for processing numerical data. However, because of the influence of graph theory (GT), there is currently a discernible movement toward non-numerical techniques.

Isomerism in the Constitution and Steric

The vertex degree is the number of lines that meet at a vertex in graph theory. Regular graphs are those in which the degrees of all the vertices are equal. 128 years ago, the mathematician Cayley put forth an algorithm to determine the number of constitutional isomers for alkanes, C_nH_{2n+2} and alkyl derivatives, like $C_nH_{2n+1}Cl$. Here, carbon atoms are represented by the n vertices of degree 4, and hydrogen atoms are represented by the 2n + 2 vertices of degree 1.

The equivalent set of trees with n vertices of degrees 1 to 4 can be considered instead, whose edges only indicate C-C bonds. This option ignores all hydrogen atoms. The hydrogen-depleted or skeleton graph of the molecule is made up of these edges and its vertices.

Data Mining

The process of collecting useful information from massive databases using techniques from database systems, statistics, artificial intelligence, and machine learning is known as data mining. Finding patterns in these enormous datasets is the primary goal. Typical data mining jobs consist of:

1. Anomaly detection: Finding odd data records that might indicate mistakes or important information.
2. Dependency or Learning by Association Rules Modeling: Determining how variables relate to one another.
3. Clustering: Putting related items in one group.
4. Classification: Using established structures on fresh data.
5. Regression: Finding functions that correspond to data with the least amount of error.
6. Summarization: Producing condensed dataset representations in the form of visualizations and reports.

Data mining relies heavily on clustering analysis, where clustering algorithms group objects according to shared attributes. During the clustering process, clusters automatically allocate objects to create classes. Clustering, sometimes referred to as "unsupervised learning," does not necessitate prior knowledge of class labels. In data mining, a variety of clustering methods are available, which group things with similar characteristics together and dissimilar objects in different groups.

Graph theory and algorithms

1. The Dijkstra's Shortest Path Algorithm:

Dijkstra's algorithm is applied to graphs with non-negative weights. The fundamental concept behind this algorithm is to maintain a list of vertices that have been explored so far, and at each stage, select the vertex closest to the source in terms of path weight. It keeps track of the shortest path weight

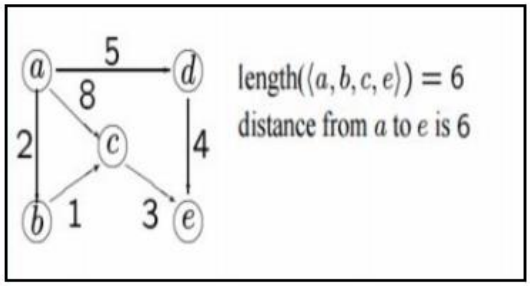
from the source for each vertex.

At each iteration, the algorithm examines all outgoing edges e=(v,u) from the current vertex v. When processing vertex v, it considers the new potential path to vertex u by adding the edge (v,u). The algorithm then updates the shortest path weight for vertex u based on this new path.

Formally, let G=(V,E) represent a weighted directed graph, where E consists of edges with real-valued weights. If e=(u,v), then w(u,v) indicates the weight of the edge. The length of a path $p=(v_1,v_2,...,v_k)$ is the sum of the weights of its edges:

$$\text{Length}(p)=\sum_{i=1}^{k-1} w=(v_i, v_{i+1})$$

The distance from vertex u to vertex v, denoted as δ(u,v), represents the length of the shortest path if one exists; otherwise, it is considered undefined, indicating no path is available. This algorithm is commonly used in applications like routing and network optimization to find the most efficient path.

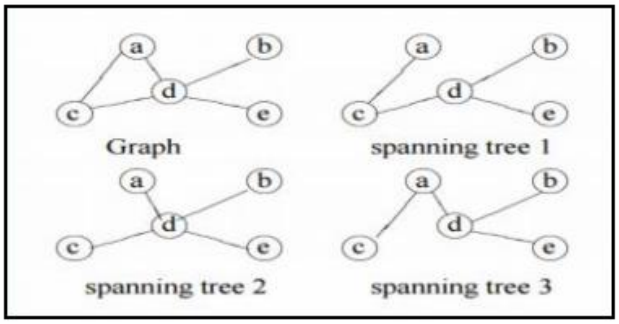


Graph 1

2. Minimum spanning tree:

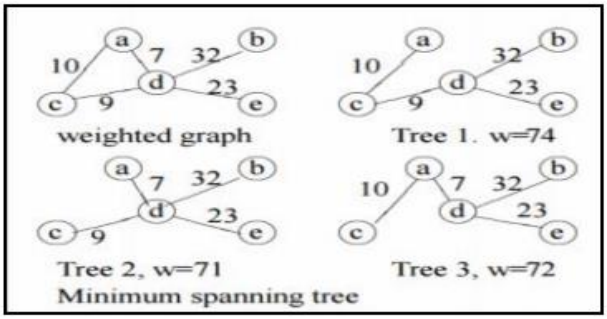
A spanning tree is a subgraph of a graph that joins all of the graph's vertices to form a tree. Weights can be assigned to each edge in a connected, undirected graph to indicate how "unfavorable" or expensive the edge is. A spanning tree's weight is the total of the weights of its included edges.

For an undirected graph G = (V, E), a spanning tree T is a subgraph that contains all of the vertices of G and is also a tree. As shown in Graph 2, a minimum spanning tree (MST), also known as a minimum weight spanning tree, is a spanning tree whose total weight is less than or equal the total weight of any other spanning tree in the graph. The objective of determining a minimum spanning tree is to connect every vertex in the graph while minimizing the tree's overall cost.



Graph 2

As seen in Graph 3, a Minimum Spanning Tree (MST) is a spanning tree with the lowest overall weight relative to all other spanning trees in an undirected, linked, weighted graph.



Graph 3

3. Determine the Planarity of the Graph:

Planarity testing, as used in graph theory, is the process of determining whether a given graph is planar, or if it can be depicted on a plane without edge intersections. The majority of planarity test methods are asymptotically optimal since they run in O(n) time, where n is the number of edges (or vertices) in the graph. Planarity can be tested using Kuratowski's theorem, which states that a graph is planar if and only if it doesn't contain a subgraph that is a subdivision of either K_5 which is the complete graph on five vertices, or $K_{3,3}$ which is the complete bipartite graph on six vertices, with three vertices connected to each of the other three.

Kuratowski's Theorem in Planarity Testing

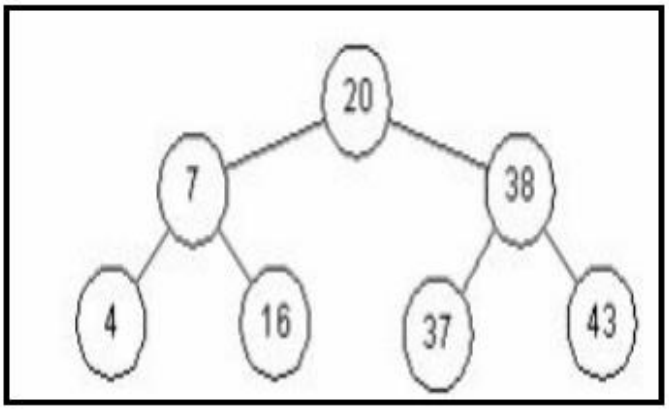
To test for a K_5 subdivision, select five vertices from the graph G and verify whether all five vertices are connected by 10 distinct paths (calculated as $5\times4 = 10$).

$$2$$

Similarly, to test for a $K_{3,3}$ subdivision, select six vertices from G and check if they are connected by 9 distinct paths (since $3\times3=9$).
Both of these methods involve exponential time complexity, making them inefficient for large graphs.

4. Data Structure elements can be found using these algorithms:

Any tree structure with left and right children for every node is called a binary search tree. It is possible for one or both of these kids to be absent. Consider k to be the value of a certain node in a binary search tree. The tree demonstrates the following characteristic: all children to a node's right have values bigger than k, while all children to the node's left have values smaller than k. The tree's exposed nodes at the base are termed leaves, while the top node is called the root. In the picture below, for instance, node 20 represents the base, whereas nodes 4, 16, 37, and 43 represent the leaves. The longest path from a tree's root to a leaf is defined as the tree's height. Graph 4 illustrates that the tree's height in this instance is 2.



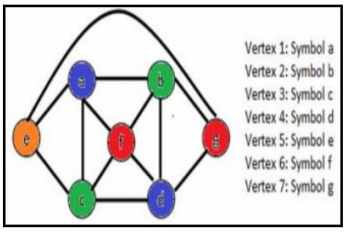
Graph 4

In a tree, we start at the root and work our way down to find a certain value. For example, we start by noting that 16 is less than 20 before moving on to the left child in order to look for 16. Since 16 is greater than 7, as shown by the second comparison, we can move on to the next child.

Graph Coloring

Graph coloring is the process of giving a graph's vertices colors or other distinguishing identifiers. According to strict definitions, a coloring is correct if no two neighboring vertices have the same color. The mapping $c{:}V(G)\rightarrow S$ c:V(G)→S, where the elements of S S denote the colors, is a (vertex) coloring of a graph G. When $|S\prec = k|$S|=k, we designate c as a k-coloring. An appropriate coloring is one in which every neighboring vertex has a different color.

The ability to appropriately color a graph with k colors indicates that it is k-colorable. In order for G to be k-colorable, the chromatic number χ(G) is the least k. It is evident that $\chi(G)$ exists since a valid $|V(G)|$ -coloring is produced when unique colors are assigned to vertices. The best coloring for G is a $\chi(G)$ -coloring. If g(G)=k, then a graph G is k-chromatic. Each class of color creates a stable set when colored properly. Consequently, a k-coloring may alternatively be thought of as a division of the vertex set of G into k disjoint stable sets $Si= \{v \mid c(v)=i\}$ for $1\leq i\leq k$ as illustrated in Graph 5.



Graph 5

Applications in Scheduling

Graph theory is essential in scheduling, a common application in computer science. For example:

1. Aircraft Scheduling:

The following is an approximate description of aircraft scheduling:

- A collection of planes and flights are the input.
- Output: Individual aircraft rosters that guarantee all flights are assigned.

The aircraft type for each flight has not yet been decided, but the flight schedule, including the times of departure and arrival, is set at the beginning of the scheduling process. The anticipated number of passengers and the flight distance are two examples of the factors that influence the kind of aircraft chosen for each voyage. For example, a Boeing 767 can travel farther (such as over the Atlantic Ocean) and carry twice as many passengers as a DC9.

Generally speaking, the DC9 would be used for short-haul flights and the 767 for long-haul ones. However, a 767 might be needed for short-haul flights with a large passenger load. As a result, aircraft types are taken into consideration during the timetable preparation process, but they are only

partially restricted and not fixed. The precise aircraft type for each flight is decided upon during the aircraft scheduling process.

2. Scheduling of Jobs:

In computers, scheduling is the process of allocating resources to finish tasks that have been specified in some way. These resources could be virtual computing components like threads, processes, or data flows that are scheduled onto hardware components like extension cards, processors, or network lines. Schedulers are frequently used to accomplish a desired quality of service, enable efficient resource sharing among different users, or keep all compute resources occupied (as in load balancing). Scheduling is an essential component of a language's execution paradigm and is important to computation itself. Multitasking is made feasible by a scheduler, which allows multiple processes to run simultaneously on a single processor.

Throughput maximization (the amount of work completed per time unit), response time minimization (the amount of time between when work is enabled and when it begins to execute on resources), latency minimization (the amount of time between when work is enabled and when it is completed), and fairness maximization (making sure each process receives an equal amount of CPU time) are some of the objectives that are achieved through scheduling. Generally speaking, these objectives frequently clash, for example, throughput versus delay.

Applications of Graph Theory

1. Computer Science:

Graphs are used in computer science to depict communication networks, data organization, computing devices, computation flow, etc. For example, a directed graph, where the vertices represent web pages and the directed edges indicate links between pages, can be used to depict the link structure of a website. By collecting statistics on graph-theoretic features associated with the atoms' topology, condensed matter physicists can quantitatively study the three-dimensional structure of complex simulated atomic formations. With vertices standing in for atoms and edges for bonds, a graph is a natural representation of a molecule in chemistry. This method works particularly well for computer processing of molecular structures, from database searching to chemical editors.

2. Chemistry:

Graphs can depict the dynamics of a physical process on a system or the local relationships between interacting components of a system in statistical physics. The micro-scale channels of porous media are also represented by graphs, where the vertices stand in for the pores and the edges for the smaller channels that connect them. A vertex might represent a location where a certain species (or habitat) exists, and the edges can indicate migration routes or movement between the regions. This is another way that graph theory is useful in biology and conservation efforts. This data is crucial for researching breeding trends, monitoring the transmission of parasites or diseases, or determining how shifts in mobility may impact other species.

Graph-Based Method for Classifying Fingerprints

These methods are predicated on the finding that relational graphs are not affected by rotations of fingerprint images. As a result, it seems that the relational graph can be used to solve the rotation problem in fingerprint representations. The issue is only partially resolved, though, because each graph node must be enhanced with a set of attributes that are typically reliant on fingerprint rotation. Segmenting the orientation field into sections with homogeneous directions is the foundation of the initial concepts for creating graph-based representations of fingerprints. Each zone is represented by a node in the graph, and two nodes representing neighboring regions are connected by an edge.

In order to find the best match for final classification, Lumini et al. have adopted this concept and use inexact graph matching with a template graph for each class. Finding a group of graph prototypes is quite challenging because of the considerable variety of the segmentations that are obtained. To represent as many variants as feasible, a lot of prototypes are required, but the high computational cost of graph matching techniques merely lengthens the classification time.

In keeping with Lumini et al.'s concept, Cappelli et al. identified a collection of prototype segmentations for every class and created an algorithm to "guide" the orientation field segmentation, resulting in a segmentation that is reliant on the class. Every classification comes with a price. Five dynamic masks are used to segment an input fingerprint's orientation field. The fingerprint is linked to the relative class, and the least-cost segmentation is the "structure" that most accurately depicts a certain orientation field.

However, due to the modest between-class difference, identified prototypes are not the best for efficiently differentiating between fingerprint classes.

Conclusion

Graph theory applications and related research difficulties were reviewed in this work. X-ray crystallography, radar, astronomy, circuit design, coding theory, algorithms, computation, biology, chemistry, communication networks, coding theory, coding theory, and database administration are among the subjects discussed. Many issues still need to be resolved, even with the tremendous advancements in these fields. It is hoped that this review will encourage new researchers to delve into the wide possibilities of graph theory and help the subject progress.

***Department of Mathematics**
S.B.R.M Govt. College
Nagaur (Raj.)

References

1. Shrinivas.S.G, Vetrivel.S, Elango.N.M. International Journal of Engineering Science and Technology Vol. 2(9), 2010, 4610-4621.
2. Beratta, E.Vetrano, F.Solimano, F.Lazzari, C. Bull Math.Bid.1979, 41, 641.

3. King, R. B. Theor. Chim. Acta 1980, 56, 269.
4. Fayyad,U. Shapiro,G.P. and Smyth,P. "From data mining to knowledge discovery in Databases," AI Magazine, 1996, 37-53.
5. Clarke, B. L. Stud. Phys. Theor. Chem. 1983,28, 322.
6. Narasingh Deo, "Graph theory with applications to engineering and computer science", Prentice Hall of India, 1990.
7. Balaban, A.T. Rouvray, D. H. In "Applications of Graph Theory"; Wilson. R. J.: Beinecke. L. W. Eds.: Academic Press: London. 1976.
8. Biggs, N. L.; Lloyd, E. K.; Wilson, R. J. "Graph Theory 1736-1936."
9. Balaban, A. T., Ed. "Chemical Applications of Graph Theory"; Academic Press: London, 1967.
10. Lumini, A. Maio, D. and Maltoni, D. Inexact graph matching for Fingerprint Classification, Machine Graphics and Vision, 1999, 8 (2) 241-248.
11. Rouvray, D. H. Chem. Soc. Rev. 1974, 3, 355.
12. Jain, A.K. Murty, M.N. and P. J. Flynn, "Data clustering: A review," ACM Computing Surveys (CSUR), 1999, 31(3), 264-323.
13. King, R. B., Ed. "Chemical Applications of Topology and Graph Theory". Stud. Phys. Theor. Chem. 1983, 23.
14. Cayley, A. Philos. Mag. 1857, 13 (1), 172.
15. http://www.cse.ust.hk/~dekai/271/notes/L07/L07.pdf.
16. http://www.tcs.tifr.res.in/~workshop/nitp_igga/slides/shreesh-planarity-patna.pdf.
17. http://www.cse.ust.hk/~dekai/271/notes/L10/L10.pdf.
18. https://www.cse.ust.hk/~dekai/271/notes/L07/L07.pdf.
19. https://en.wikipedia.org/wiki/Graph_theory#cite_ref-1.
20. http://epaperpress.com.
21. Cappelli, R. Lumini, A. Maio, D. and Maltoni, D. Fingerprint Classification by Directional Image Partitioning, IEEE Trans. on PAMI, 1999, 21 (5) 402-421.
22. Glass, L. J. Chem. Phys. 1975, 63, 1325.
23. Balaban. A. T. Math. Chem. 1975. 1. 33.