International Journal of Computer and Communication Technology

Volume 1 | Issue 3 Article 3

July 2010

Studies on Dimultigraph and Prograph Based Applications of Graph Theory in Computer Science

Biswajit Bhowmik
Department of Computer Science & Engineering Bengal College of Engineering & Technology Durgapur – 713 212 India, biswajitbhowmik@gmail.com

Studies on Dimultigraph and Prograph Based Applications of Graph Theory in Computer Science

Biswajit Bhowmik
Sr. Lecturer
Department of Computer Science & Engineering
Bengal College of Engineering & Technology
Durgapur – 713 212
India
Email: biswajitbhowmik@gmail.com

Abstract: Graph Theory has gained an impulsion in the past few years in rapacious dimensions. Day by day with rapid advancements of the technology, its demand for proper utilization of resources is increasing at a drastic rate. Simultaneously, it has emerged as one of the most powerful tools for the illustration and solution of the problems that are basically chronological in nature. And many seemingly diverse problems in computer science have been worked out with the help of this thought. A special approach (DPBA model) for different kinds of graphs and their applications in computer science is proposed in this paper. It introduces dimultigraph by considering digraph. The approaches of dimultigraph are enumerated in simple manner. Introduction of prograph in performance evaluation in the model reflects its freshness as well as widely acceptance both theoretically and in implementation.

Keywords: Digraph, Dimultigraph, Prograph, DFD, OOSE.

INTRODUCTION

The study of graphs, graph theory, both in mathematics and computer science models pairwise relations between objects (vertices) from a certain collection with a collection of edges.

Definition: A graph G = (V, E) can be defined as sets of $V = (v_1, v_2 \ldots v_n)$ called vertices and $E = (e_1, e_2 \ldots e_m)$ called edges. Here every edge $e_k \in E$ is an unordered pair $(v_i, v_j) \in V$. The vertices v_i, v_j associated with an edge e_k are its end vertices [6]. This G is called undirected graph shown in Fig. 1.

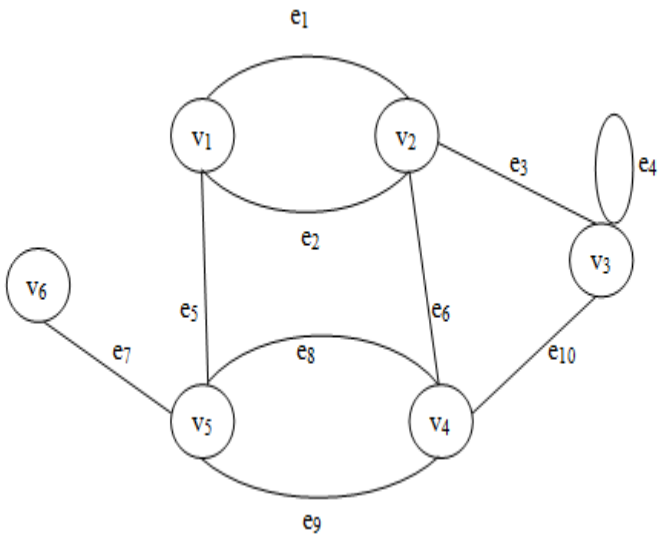

Fig. 1: A graph with six vertices and ten edges.

This simple definition puts Graph Theory together with some following Basic Terminologies [6, 7].

A. Digraph

A graph G = (V, E) is said to be a directed graph or digraph if V is a set of vertices and E is a set of ordered pair of vertices from V. Fig. 2 is a digraph.

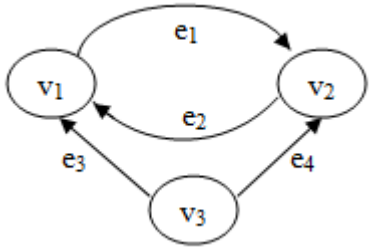

Fig. 2: A digraph with three vertices and four edges.

B. Dimultigraph

A digraph G = (V, E) is said directed multigraph (Dimultigraph) iff V = a nonempty finite set of vertices and E = a finite multiset set of ordered pairs of distinct elements of V called edges. Thus, a multigraph allows multiple edges between two vertices [4].

C. Initial Vertex

The vertex from which an edge is incident out of a graph is called initial vertex. E.g. v_1 is the initial vertex for the edge e_1 in Fig. 2.

D. Terminal Vertex

The vertex on which an edge is incident into in a graph is called terminal vertex. E.g. v_1 is the terminal vertex for the edge e_2 in Fig. 2.

E. Adjacent Vertex

When two vertices are connected by an edge then they are said to be adjacent to each other. E.g. in Fig. 1 the vertices v_5 and v_6 are adjacent to each other as they are linked with the edge e_7.

F. Self Loop

If an edge having same initial and terminal vertex in a graph is called self loop. E.g. the edge e_4 is a self loop in Fig. 1.

G. Parallel Edge

If two or more edges have originated from a vertex and incident on another vertex then edges are said to be parallel edges. Parallel edges thus have common pair of vertices. E.g. in Fig. 1 the edges pairs (e_1, e_2) and (e_8, e_9) are parallel edges separately. But the edges (e_1, e_2) in Fig. 2 are not parallel edges.

H. Simple Graph

A graph without any self loop or parallel edge is called a simple graph. E.g. the digraph in Fig. 2 is a simple digraph.

I. Complete Graph

A graph is said to be complete graph if all the vertices in the graph are adjacent to each other.

J. Path

A path in graphs is a sequence of edges such that no vertices appear more than once in the sequence. E.g. $e_3 - e_1$ forms a path in Fig. 2.

K. Connected Graph

If there is a path between any pair of vertices in a graph then it is called a connected graph. E.g. Fig. 1 is a connected graph.

L. Cycle

The path containing two or more edges starting and ending at the same vertex is called cycle of the graph.

E.g. the edges pair (e_1, e_2) in Fig. 2 forms a cycle.

M. Cyclic Graph

A graph containing at least one cycle is called cyclic graph otherwise an acyclic graph. Both the graphs in Fig. 1 and in Fig. 2 are cyclic graph.

N. Subgraph

A graph g = (V', E') is said to be sub graph of a graph G = (V, E) iff E' $\in$ E and V' $\in$ V. Similarly E' and V' are said subset of E and V respectively.

GRAPH REPRESENTATIONS IN COMPUTER

Depending on problems, machine type, language, etc a graph is stored in digital computer using any one of the following approaches:

a. Adjacency matrix.
b. Incidence matrix.
c. Edge listing.
d. Two linear arrays.
e. Successor listing. Etc.

Adjacency Matrix

The most accepted practice that feeds a graph/digraph G having n vertices to computer is adjacency matrix. This is basically an nxn binary (0/1) matrix and is defined as.

$$\text{Adj}(i, j) = \begin{cases} 0, \text{ if the edge } (i,j) \in E \\ 1, \text{ otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Obviously, Adj(i, j) requires n^2 bits of computer memory. If w be the computer word length, each row of the matrix may possibly be written as n bits sequence in $\lceil n/w \rceil$ machine words. Thus computer memory required to store Adj(i, j) is $n\lceil n/w \rceil$ words. Due to symmetric in nature, adjacency matrix of an undirected graph with n vertices needs n(n-1)/2 bits storage only [6, 8].

Incidence Matrix

Sometimes a graph G with n vertices and e edges is stored and manipulated using incidence matrix. Like Adj(i, j) this is also a binary matrix and is defined as.

$$\text{I}(i, j) = \begin{cases} 1, \text{ if } j\text{th edge is incidence on} \\ \quad i\text{th vertex} \\ 0, \text{ otherwise} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots (2)$$

Generally, number of edges (|E|) is greater than number of vertices (|V|) i.e. |E| > |V|, storage bits for I, n.e > n^2 [6, 8].

III. WHY GRAPHS?

At the beginning whatever may be the problem, it is viewed as a set of sub problems, events or states (nodes/vertices in a graph) and the transitions (edge in a graph) between these events [2]. Consequently, graphs are extremely important in Computer Science mainly for the reasons:

a. Without graphs, representations of many problems in computer science become abstract.
b. The solution to a problem as a graph may derive new approaches directly.
c. Many problems involve representing relationships between objects, places, or concepts [4].

IV. APPLICATIONS OF GRAPHS IN COMPUTER SCIENCE

Graph Theory with application of suitable graphs nowadays turns into a major device to problems of computer science. The versatility of the graphs is really appreciated by exposure to a wide variety of applications. In this section, a list of major applications has been chosen.

Graphs in Computer Network

A set of devices (computers) connected through communication links (wires) forms computer network. Graphs in this case are utilized in defining, representing, classifying the physical layout of a network with optimal cost which varies with wire length. It is also used in categorizing the network in LAN (Local Area Network), MAN (Metropolitan Area Network), and WAN (Wide Area Network) to determine its size (covered locality) [12]. In all the cases simple graphs are often used. E.g. the following Fig. 3 is a graph of a "real world" LAN [4].

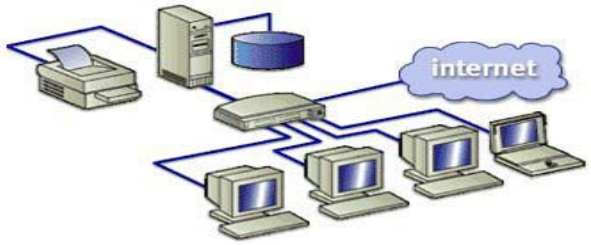

Fig. 3: Typical Computer Network using LAN.

Not only that a Dag (digraph without any cycle) is employed in information categorization system as the folder in a computer network. All of these, shortest path problems (especially single source and all pairs) are extensively consumed in routing problems. We solve these problems using known Dijkstra's, Bellman-Ford's, Flyod-Warshall's algorithms. E.g. problems of efficiently planning routes for mail delivery, garbage pickup, snow removal, diagnostics a computer in networks, and others, can be solved using models that involve effective or least cost paths in graphs [5, 13].

Graphs in Computer Network Security

Recently computer scientists are employing the vertex cover algorithm to simulate stealth worms' propagation on large computer networks and propose optimal strategies against such attacks for protecting the network.

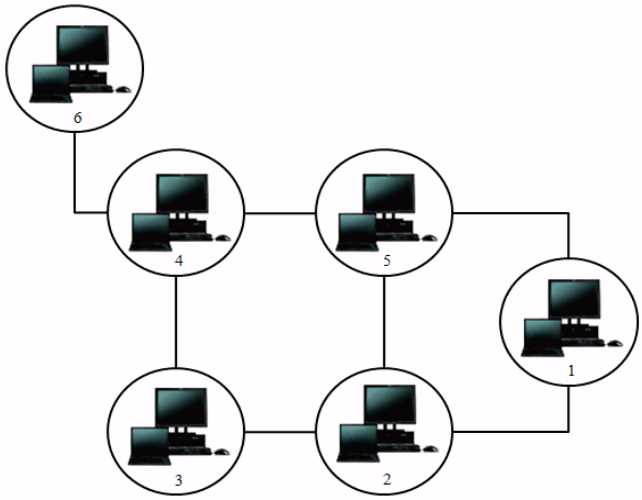

Fig. 4: Minimum vertex cover set {2, 4, 5} of the computer network.

The idea is to find a minimum vertex cover in the graph (network) whose vertices are the routing servers and edges connect them. This idea has now been become an optimal solution for worm propagation and designing the network defense strategy. Fig. 4 shows a simple computer network and a corresponding minimum vertex cover {2, 4, 5} [3].

Graphs in Data Transmission

A message constituted of n symbols for communication between sender and receiver would be competent through least transmission time as well as less amount of storage space. An elegant solution to the relevance was discovered by D. Huffman who designed Huffman algorithm (to honour him) and explained with minimum weighted external path length binary tree. He proved that number of bits r required sending n symbols can be determined by the inequality.

$$2^{r-1} < n \leq 2^r \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Graphs in Coding Theory

Gray codes (also called circuit code, cyclic code, or reflected binary code), so important in analog to digital conversation of information, BCD, needs change in more bits, etc are represented by cubic structures. They are used to symbolize number system. An m-bit Gray code corresponds to a circuit in an m-cube having 2^m vertices utmost. When all these vertices are used by m-bit code, then it is called complete code. It does not need to be a circuit code. E.g. decimal digits can be denoted by using 10 out of 16 vertices for 4-bit words [6].

Graphs in Web design

The link structure of a website could be represented by a directed graph: the vertices are the available web pages and the link from page A to page B is the directed edge. A similar approach can also be taken to problems computer chip design [1].

Graphs in Finite Automata

Finite automata (finite state machine) employs digraph to represent state transitions, its classification – NFA and DFA, etc. This kind of diagram is called state transition diagram. Fig. 5 is a NFA [4].

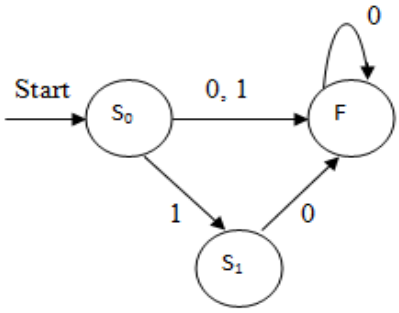

Fig. 5: State diagram for NFA.

Graphs in Computer Programming

Graphs are playing an increasingly significant role in the design, analysis, and testing of computer programs. The purpose could be subdividing a large program into a number of subprograms, estimation of the running time and/or storage requirement, detecting errors (structural), documentation, or simply understanding the code written by someone else. For all the reasons, it is handy to express a program as a digraph. The graph thus obtained might be termed as *program digraph* (Prograph). Each vertex v_i of the prograph is a program block – each having one entry (first instruction in block) and one exit point (last instruction in block). Each edge (v_i , v_j) represents flow of control from the exit point of v_i to the entry point of v_j. Fig. 6 shows a typical prograph.

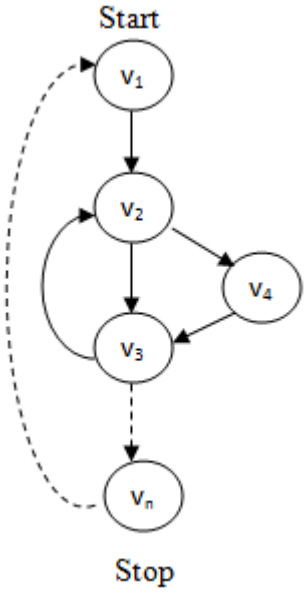

Fig. 6: A typical prograph of n nodes.

A prograph can also be thought as an abstraction of a flow chart. One derived graph known as a cyclomatic tree is of particular value in program testing. It is so named because the

number of leaves of the tree is equal to the cyclomatic number of the program graph [11, 6].

Graphs in Software Engineering

Few years ago software design which consists of a set of modules, sat at the kernel of software engineering. Before implementation, an architectural design is made. Each module in the design provides its necessary information and their links supply information sharing sequence at different levels. The notation we use is called Data Flow Diagram (DFD). But in recent times Object-Oriented Software Engineering (OOSE) has been employing classes as modules to deal with the analysis, design and implementation of systems. The latter field can greatly be benefited from the application of Graph Theory, since the main mode of representation, namely the class diagram, is essentially a directed graph [10].

V. CONCLUSION

The novelty of this model (DPBA) not only provides various application areas at a glance but also serves as platform to implement the approaches with satisfactory results. A part of the measureless applications has been given you an idea about. Further study on extending this model with more expectations is in progress. Furthermore, Graph Theory cannot be limited its applications only in computer science rather obviously takes steps as a key mode of analysis miscellaneous problems in various areas like in mathematical research, communications, electrical engineering, sociology, economics, marketing, business administration, and so on. Later on we will focus on the significance of this model in above mentioned areas.

REFERENCES

[1] www.en.wikipedia.org/wiki/Graph_theory.
[2] William S. Bowie, "Applications of Graph Theory in Computer Systems", International Journal of Parallel Programming, Volume 5, Number 1 / March, 1976, pages 9-31.
[3] www.dharwadker.org/pirzada/applications.
[4] Dai Tri Man Le, "Lab 10 - Graph Theory Computer Science 1FC3".
[5] www.aix1.uottawa.ca/~jkhoury/graph.htm
[6] N. Deo, "Graph Theory with Applications to Engineering and Computer Science", PHI, 23rd edition, 2002.
[7] B. Bhowmik, "Design and Analysis of Algorithm", WBUT series, chapter 11, first edition [in press].
[8] N. Guruprasad, "Data Structures using C", Schitech, 2nd Edition.
[9] D. Samanta, "Classic Data Structures", PHI, 2nd reprint, 2002.
[10] Rogger S. Pressman, "Software Engineering – A Practioner's Approach", MGH, 5th Edition, 2001.
[11] www.portal.acm.org/citation.cfm?id=811109.
[12] Behrouza A. Forouzan, "Data Communications and Networking", MGH, 4th Edition, 2008.
[13] E. Horowitz, S. Sahni, S. Rajasekharan, "Computer Algorithms/C++", Universities Press, 2nd Edition, 2008.