# CHAPTER 1

# INTRODUCTION

## 1.1 GRAPH ALGORITHMS AND PARALLEL COMPUTATION

Graphs are essential tools for understanding and solving problems in diverse disciplines. For example, graph theoretic models are convenient for network analysis in electrical engineering, planning of schedules in operations research, identification of chemical compounds in organic chemistry, analysis of Markov chains in probability theory, organization of data structures in computer science. In fact, applications of graphs have a very wide range (see, e.g., [167,177 ]) and cannot probably be listed exhaustively. Graphs arising out of real-life problems, are often very large and cannot be analyzed without the aid of computers. Naturally efficient algorithms for graph theoretic problems are of immense practical importance.

In order to answer almost any non-trivial question about a graph, every arc and in the process every node of the graph must be examined at least once. For example, in order to determine whether or not a graph is connected, every arc of the graph has to be checked. In case the graph is declared disconnected ignoring an arc, it is quite likely that the ignored arc might have made the graph connected. The same can be said about the questions concerning separability, planarity and the like.

There are two important ways to search the arcs of a graph viz., depth-first search and breadth-first search. The graph search techniques have been found to be useful for algorithmic solutions of a variety of important and interesting graph theoretic problems. The concept of depth-first search has been used since the nineteenth century (see [ 41 ] ). However, it was formalized first by Hopcroft and Tarjan [ 74 ] . Breadth-first search seems to have been used first by Moore [ 113 ] . Another way to search a graph is by breadth-depth search. Breadth-depth search has been used extensively in order to develop efficient serial algorithms for the shortest-path problems and PERT networks.

A search of a graph can be speeded up if more than one arc are examined at a time. In general, if some or all the steps of a desired computation are executed in parallel, greater execution speed is achieved. Parallel processing, therefore, provides a natural solution to the problem of meeting the ever increasing demand for the computing power of computers. Parallel algorithms have been studied since early sixties although no parallel computers had been built at that time. Greater interest in parallel algorithms has been generated due to the emergence of highly parallel computer systems like ILLIAC IV[ 8] , CDC-STAR[69] , STARAN [10] , Cmmp[ 179] , Cm* [157] etc.

A shared memory model of a Single Instruction-stream Multiple Data-stream (SIMD) computer is the most widely used model of parallel computation for theoretical studies (see e.g., [4,34, 173, 180 ]). This model consists of a master processor, a number of slave processors and an unbounded global memory. All the slave processors have substantial local memories and can access the global memory. The master processor broadcasts instructions to the slave processors. An instruction of the master processor is executed by all the active slave processors simultaneously and synchronously over different data. More than one slave processor are allowed to read from the same location of the global memory at the same time. Usually no two slave processors are allowed to write simultaneously on the same location of the global memory. When different processors are allowed to write on the same location of the global memory, the model is said to suffer from memory store conflicts.

A number of efficient parallel algorithms have been developed for wide assortments of graph theoretic problems on a shared memory model of an SIMD computer [4,34,35,57,71,137,144, 180 ] . The first attempt to design parallel algorithms for graph search on a shared memory model of an SIMD computer was made by Reghbati and Corneil[ 137] , but they remarked that depth-first search is unsuitable for processing graphs in parallel. Later, Eckstein [34] proposed parallel algorithms for depth-first search and breadth-first search of graphs. She also developed parallel algorithms for some graph connectivity problems using depth-first search and breadth-first search. Eckstein's study indicated that depth-first search can be used effectively for processing non-sparse graphs in parallel.

## 1.2 MOTIVATION OF THE PRESENT WORK

The present study embodies the results of an investigation on parallel processing of graphs. The importance of parallel processing of graphs and the general belief that graph search methods are unsuitable for parallel processing of graphs are the two major reasons which have motivated this investigation. The results obtained are interesting enough to suggest that graph search can be used effectively for parallel processing of graphs, both sparse and non-sparse.

The simplest type of graph is a tree. Trees, besides being the most important nonlinear structures used in computer algorithms, have many other applications (see [87]). So it is often required to traverse a tree by visiting i.e., processing its nodes in some systematic order. The visit at a node may be as simple as printing its contents or as complicated as computing a function. There are many ways to traverse a tree. Out of these the preorder, the postorder and the breadth-first order are frequently used to traverse a tree. In addition, the reverse postorder is also found to be useful. However, when an algorithm for the postorder traversal is available the reverse postorder traversal is accomplished easily effecting only minor modifications of the postorder traversal algorithm. Wyllie[180] was the first to suggest parallel algorithms for traversals of binary trees on a shared memory model of an SIMD computer.

Directed acyclic graphs belong to a class of graphs that are more complicated than trees. Directed acyclic graphs are almost invariably used to represent precedence relations among individuals or objects of a collection. Consequently, directed acyclic graphs find applications in a wide assortment of problems. It may be noted that directed acyclic graphs are far less complicated topological structures than the general graphs. So it is expected that there are more efficient and simple search algorithms for directed acyclic graphs than those available for general graphs. Of course, the best serial algorithms for depth-first search, breadth-first search and breadth-depth search are optimal irrespective of the complexity of the graph being searched.

Graphs arising out of real-life problems often have cycles. So it is important to evolve search procedures for general graphs. There are optimal serial algorithms for depth-first search, breadth-first search and breadth-depth search of graphs. Reghbati and Corneil [137] proposed parallel algorithms for breadth-first search and breadth-depth search of graphs. Eckstein [34,35 ] developed parallel algorithms for depth-first search and breadth-first search of graphs. All these parallel search algorithms work on a shared memory model of an SIMD computer.

An important set of problems dealing with connectivity of graphs requires finding paths between all pairs of nodes of a graph (see [2 ]). A frequently occurring path finding problem is the shortest-path problem. In the shortest-path problem it is required to find a path between each pair of nodes of an arc weighted graph such that the weighted length of this path is less than or equal to that of any other path between the pair of nodes. Dekel, Nassimi and Sahni [24] developed efficient parallel algorithms for matrix multiplication on cube-connected and perfect-shuffle computers. They showed that by using matrix multiplication algorithms the minimum of the weighted lengths of all paths between each pair of nodes can be obtained in $O((\log n)^2)$ time when $O(n^3)$ processors are used. However, this method cannot identify a minimum weighted path i.e., shortest-path between a pair of nodes.

The cycles of a graph give information as to how well it is connected. A fundamental set of cycles forms a basis for the cycle space of a graph. Therefore, the problem of finding a fundamental set of cycles is an important graph connectivity problem. The removal of a bridge increases the number of connected components of a graph by one. Thus the problem of locating bridges of a graph, is another important graph connectivity problem. There are optimal serial algorithms for the above mentioned problems (see [138,162]). Savage and Ja' Ja' [144] proposed efficient parallel algorithms for finding a fundamental set of cycles and for locating the bridges of a graph on a shared memory model of an SIMD computer. Another related graph connectivity problem is to orient the arcs of a connected, bridgeless undirected graph so that the resulting directed graph is strongly connected. There is a very simple serial algorithm for this problem that uses depth-first search. Atallah [6 ] and Vishkin[174 ] have developed parallel algorithms for orienting the arcs of a connected, bridgeless undirected graph in order to make it strongly connected. These strong orientation algorithms work on a shared memory model of an SIMD computer. The model assumed by Vishkin, allows memory store conflicts.

## 1.3 AN OVERVIEW OF THE THESIS

The present work is centered around designing parallel algorithms for tree traversals, graph search and graph connectivity problems on a shared memory model of an SIMD computer. Parallel algorithms are proposed for the preorder, the postorder, the reverse postorder and the breadth-first order traversals of trees. A parallel algorithm for depth-first search of a directed acyclic graph is developed using these tree traversal algorithms. The above mentioned algorithm for depth-first search is asymptotically faster than the parallel depth-first search algorithms proposed earlier by other workers. Parallel algorithms are developed for breadth-first search and breadth-depth search of a general graph. Each of these search algorithms is asymptotically faster than the corresponding parallel search algorithm proposed previously. Parallel algorithms are proposed for graph connectivity problems like identifying a shortest-path between each pair of nodes of a positively arc weighted graph, finding a fundamental set of cycles and the bridges of an undirected graph and strongly orienting a connected, bridgeless undirected graph. The parallel shortest-path algorithm is more efficient than those proposed earlier. Similarly, the parallel algorithms for finding a fundamental set of cycles and the bridges of an undirected graph and the parallel algorithms for strongly orienting a connected, bridgeless undirected graph are more efficient than the corresponding parallel algorithms proposed by other workers, when the underlying graph is sparse.

This thesis is organized into nine chapters including the present introductory chapter.

Chapter 2 contains a survey of relevant literature. A brief review of serial graph algorithms which use one of the graph search techniques viz., depth-first search, breadth-first search, is given at the beginning. The remaining portion of this chapter is devoted to a state-of-the-art review of parallel algorithms.

Chapter 3 presents graph theoretical terminology and notations used in this thesis. This chapter contains brief outlines of serial algorithms for important tree traversals and graph search. Besides these, it contains a discussion on the model of parallel computation used in this thesis.

Chapter 4 describes parallel algorithms for the preorder, the postorder, the reverse postorder and the breadth-first order traversals of trees. These algorithms require a tree to be available in the form of its binary tree representation. This chapter, therefore, begins with description of a parallel algorithm that converts a given tree into its binary tree representation. The traversal algorithms have $O(\log n)$ time complexity and need $O(n^{1+\beta})$ processors when the given tree has $n$ nodes, $\beta$ being a preassigned constant satisfying $0 < \beta \leq 1$.

Chapter 5 contains an $O((\log n)^2)$ parallel algorithms for depth-first search of a directed acyclic graph having $n$ nodes when $O(n^{2.81}/\log n)$ processors are used. This algorithm produces a depth-first spanning tree of a given directed acyclic graph.

An $O(\log d. \log n)$ parallel algorithm for breadth-first search of a graph is presented in Chapter 6 where $d$ is the diameter of the graph. This algorithm produces n breadth-first spanning trees of a strongly connected graph or a connected and undirected graph. The algorithm requires $O(n^3)$ processors.

Chapter 7 contains an $O((\log n)^2)$ parallel algorithm for breadth-depth search of a graph. This algorithm produces $n$ breadth-depth spanning trees of a strongly connected graph or a connected and undirected graph. The algorithm requires $O(n^3)$ processors.

A parallel shortest-path algorithm is proposed in Chapter 8. This algorithm finds a shortest-path between each pair of nodes of a graph in $O(\log d. \log n)$ time. The algorithm requires $O(n^3)$ processors.

In Chapter 9 parallel algorithms are proposed for finding a fundamental set of cycles of an undirected graph, for locating bridges of connected and undirected graph and for strongly orienting a connected, bridgeless undirected graph. Each of these algorithms runs in $O((\log n)^2)$ time when $O(n.(m-n+1))$ processors are used. It is assumed that $m$ denotes the number of arcs of the graph. It is also indicated that these algorithms can be improved to run in $O(\log d. \log n)$ time provided $O(n^3)$ processors are employed.