

Graph and Network Algorithms

SAMIR KHULLER

Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland (samir@cs.umd.edu)

BALAJI RAGHAVACHARI

Department of Computer Science, University of Texas at Dallas (rbk@utdallas.edu)

INTRODUCTION

Graphs provide a powerful tool to model objects and relationships among objects. The study of graphs dates back to Euler's days in the 18th century, when he defined the *Königsberg bridge* problem, and since then has been pursued by many researchers. Specifically, graphs can be used to model problems in many areas such as transportation, scheduling, networks, robotics, VLSI, compilers, mathematical biology, and software engineering. Many optimization problems from these and other diverse areas can be phrased in graph-theoretic terms, leading to algorithmic questions about graphs.

Graphs. Graphs are defined by a set of vertices and a set of edges, where each edge connects two of its vertices. Graphs are further classified into directed and undirected graphs, depending on whether the edges are directed. An important subclass of directed graphs that arises in many applications, such as precedence-constrained scheduling problems, is directed acyclic graphs (DAG). Interesting subclasses of undirected graphs include trees and bipartite graphs.

Representation of Graphs. There are two main data structures for representing graphs, the *adjacency list* and the *adjacency matrix*. An adjacency list is

an array of linked lists, one for each vertex of the graph. In the linked list of a vertex, all its incident edges are stored. This representation is usually the preferred one, especially for sparse graphs, because the storage space required is linear in the size of the graph. An adjacency matrix is a 0/1 matrix in which each vertex of the graph has a corresponding row and column. An entry of the matrix is 1 if and only if there is an edge between the corresponding vertices. This representation is usually preferred when the graph is dense.

BASIC PROBLEMS

Searching Problems. Graph-searching procedures such as depth-first search (DFS) and breadth-first search (BFS) [Cormen et al. 1989; Even 1979; Tarjan 1983] form the basic preprocessing steps for most graph algorithms. Algorithms based on DFS have been known for a long time for the problem of searching mazes. However, it was the work of Hopcroft and Tarjan (for which they received the ACM Turing Award in 1986) that illustrated the full algorithmic power of DFS. They demonstrated efficient algorithms for several problems, such as finding biconnected components and bridges of a graph and testing triconnectivity and planarity. DFS on directed graphs can be used to classify its vertices into strongly connected components, to detect cycles, and

S. Khuller's research supported by NSF Research Initiation Award CCR-9307462 and an NSF CAREER Award CCR-9501355. B. Raghavachari's research is supported in part by NSF Research Initiation Award CCR-9409625.

Copyright © 1996, CRC Press.

to find a topological order of the vertices of a DAG.

Minimum-Cost Problems. Graphs can be used to model networks, where vertices model sites and edges model links between sites. Each link has an associated construction cost. Finding a set of links of minimum total cost that connects all the sites is known as the *minimum spanning tree* problem. It is known that a greedy algorithm solves the problem, and a variety of algorithms have been developed for finding minimum spanning trees. A recent exciting development was a randomized linear-time algorithm to solve the problem.

Shortest-Path Problems. A class of important problems in graphs are *shortest-path problems*, which play an important role in routing messages efficiently in networks. In this problem, each edge is associated with a weight that captures the delay in sending a message on this link. Efficient algorithms for shortest-path problems from a central vertex are known. The problems of finding the transitive closure of a graph and all-pairs shortest paths have been well studied and several algorithms have been proposed [Cormen et al. 1989; Tarjan 1983]. Efficient implementation of these basic graph algorithms led to the discovery of several data structures, such as the “union-find” structure [Tarjan 1983] and Fibonacci heaps [Cormen et al. 1989]. These data structures have been used to speed up many other algorithms.

Matchings and Flows. The concept of a matching in a graph is very fundamental and forms the cornerstone of combinatorial optimization, with an entire book devoted to it [Lovasz and Plummer 1986]. A matching in a graph is a subset of edges that have no common incident vertices. It captures the notion of pairing off compatible vertices. The maximum matching problem is that of finding a matching of maximum cardinality. Hall and Tutte gave necessary and sufficient conditions for a graph to have a perfect matching—a

matching in which all vertices are matched. Edmonds gave a polynomial-time algorithm for finding a maximum matching. Edges may be assigned weights that denote the advantage of including them in a matching. Maximum-weight bipartite matching, also known as the assignment problem, arises in many applications. Algorithms for maximum cardinality and maximum-weight matchings have been well studied. The Hungarian method is a popular technique for solving the assignment problem. See Ahuja et al. [1993], Bondy and Murty [1977], Lovasz and Plummer [1986], and Papadimitriou and Steiglitz [1982] for more details.

The maximum-flow problem is that of finding a maximum flow of a single commodity from a source vertex to a sink vertex that satisfies capacity constraints on the edges and flow conservation constraints at the vertices. Associating costs with edges yields the minimum-cost flow problem. Several graph problems, including the assignment problem and graph connectivity, can be reduced to the minimum-cost flow problem. A fundamental theorem is the *Max-flow Min-cut Theorem* of Ford and Fulkerson that shows a min-max equality between two fundamental properties of a flow network. The two major techniques for solving flow problems are the augmenting-path method and the preflow-push method. Polynomial-time algorithms to solve minimum-cost flows are also known [Ahuja et al. 1993].

HARD PROBLEMS

Many optimization problems such as the well known traveling-salesman problem are NP-hard. Polynomial-time algorithms are not known for these problems, and many researchers believe that they do not exist. Hence heuristics that produce suboptimal solutions have been a subject of much research, especially in the last decade. Other problems that arise in different contexts are the Steiner tree problem and the graph-

coloring problem. In spite of being NP-hard, large instances of the Steiner-tree problem and the traveling-salesman problem can be solved in practice, that is, solutions that are very close to optimal can be obtained. On the other hand, even small instances of the graph-coloring problem are hard to solve. Formal evidence for the hardness of the approximability of problems such as clique, independent set, and coloring was obtained recently. The complexity of graph isomorphism is a major open problem. Many other problems on graphs are known to be NP-hard (see Sections A1 and A2 in Garey and Johnson [1979]).

The Sloan Digital Sky Survey project is a real-life example of an astronomy project in which the problem of optimally placing telescopes to collect data is formulated as a graph problem and solved using minimum-cost flows.

FURTHER READING

Several textbooks describe the material mentioned in this survey. Even's [1979] book discusses many basic graph algorithms known before 1979. Tarjan's [1983] book provides a more recent survey of both algorithms and data structures. Cormen et al. [1989] have written a comprehensive text that includes most of the basic data structures and algorithms. The books by Papadimitriou and Steiglitz [1982] and Ahuja et al. [1993] provide a comprehensive coverage of combinatorial optimization. The book edited by Hochbaum [1996] provides an extensive coverage of work on approximation algorithms for graph problems.

Conferences and Current Work. Many international conferences carry articles

on recent advances in graph algorithms. Some of them are the IEEE Symposium on Foundations of Computer Science (FOCS), the ACM Symposium on Theory of Computing (STOC), the ACM-SIAM Symposium on Discrete Algorithms (SODA), the International Colloquium on Automata, Languages and Programming (ICALP), and the European Symposium on Algorithms (ESA). There are many other regional algorithms/theory conferences that carry research papers on graph algorithms. The major thrust of recent research has been in the design of approximation algorithms for graph-based optimization problems.

REFERENCES

- AHUJA, R. K., MAGNANTI, T. L., AND ORLIN, J. B. 1993. *Network Flows*. Prentice Hall, Englewood Cliffs, NJ.
- BONDY, J. A. AND MURTY, U. S. R. 1977. *Graph Theory with Applications*. American Elsevier, New York.
- CORMEN, T. H., LEISERSON, C. E., AND RIVEST, R. L. 1989. *Introduction to Algorithms*. MIT Press, Cambridge, MA.
- EVEN, S. 1979. *Graph Algorithms*. Computer Science Press, Potomac, MD.
- GAREY, M. R. AND JOHNSON, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco.
- HOCHBAUM, D. 1996. *Approximation Algorithms*. Edited volume under preparation, PWS, Boston, MA (in press).
- LOVASZ, L. AND PLUMMER, M. D. 1986. *Matching Theory*. Elsevier B.V., New York.
- PAPADIMITRIOU, C. H. AND STEIGLITZ, K. 1982. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ.
- TARJAN, R. E. 1983. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia.