**Network and Theory**

**Introduction**

Transportation networks are spatially complex and, therefore, difficult to analyze and describe. In order to make sense out of the great web of world-wide transportation systems, it is necessary to use graphs and charts to study their characteristics and effects on spatial realities. To make this approach useful, geographers purposely remove some information about a network in order to create an idealized model.

Transportation geography graph theory involves developing abstract representations of transportation networks that consist of **points and lines**. The resultant graph is not a map, and it is not an exact model of the real world. It is, instead, a conceptual representation of reality. When creating network graphs, geographers connect **points (vertices**) using **lines (edges**) in order to develop an idealized measure of the structure of actual transportation systems.

Geographer create these abstract representations of networks using **edges (linkages**) and **vertices (nodes**) in order to understand the degree to which all vertices are interconnected. This gives them a measure of regional connectivity.   Two types of measures are possible; a number that defines the cumulative geometric pattern of a network, and a trajectory of integers that measure the associations between individual features (vertices and/or edges) of a network to the total network.



**Network Theory**

Network optimization needs to use many terms and notions used in graph theory. In this chapter we seek to introduce most of the graph theory terms and notions used in the book. We also introduce some concepts used in the study of algorithms. To explain these concepts, we will make use of a fictitious map of one of the facilities of the fictitious Global Telecom Company (GTC). The map is shown in Figure B.1 and is drawn to scale.



Fig. B.1. Map of one of the facilities at GTC (to scale).

If we want to depict the connectivity among various points in the facility, we can have a schematic representation of the map as shown in Figure B.2. In this figure, the locations of the map have been replaced by points, depicted as labeled circles. In Figure B.2 for example, point 1 represents the Security building in the map, point 2 represents the Reception desk, point 3 represents Warehouse 1, and so on. The lines between the points show that the buildings represented by the points are connected by a direct road segment. Figure B.2 is known as a graph. The points 1, 2, . . . , 10 are referred to as nodes (or vertices) of the graph, and the lines between the nodes are referred to as edges (or links) in the graph. Hence a graph is just a collection of nodes and edges linking pairs of nodes. In some graphs, we allow more than one edges to connect a pair of nodes. As a special case of such graphs, an edge may connect a node to itself. Graphs with such possibilities are called multigraphs. Graphs that are not multigraphs are called simple graphs.



Fig. B.2. The map of Figure B.1 represented as a graph.

Notice that the representation that we made does not faithfully represent the geographical properties of the map; for example, the edges between 4 and 7, and 6 and 7 are drawn approximately equal, although the distance between the two production facilities in the map is distinctly more than the distance between Production Facility 1 and the Canteen. This is because the graph is just a representation that emphasizes more on the connectivity between various points rather than a faithful representation of geography.

the map, we make use of weighted graphs (see Figure B.3), also referred to as networks. Next to each edge in a weighted graph, there is a number denoting the weight (or cost, or length) of that edge. In Figure B.3, the weights denote the distances between the corresponding nodes in 100 meter units. So from it we can interpret that the distance between the Security building and the Reception desk is 200 meters.

Next let us assume that the road segments in Figure B.1 are one way roads. Let us suppose that one can go in the direction from the Security building to the Mainframe Room, but not in the other direction. The only other permissible directions of travel are from the Mainframe Room to Warehouse 2, down the straight road from Warehouse 2 to the Reception desk, from Production Facility 2 to Production Facility 1 to the Canteen, and from the Reception desk to Warehouse 1. These directional properties can be depicted in a directed graph (or digraph), which is a graph in which all edges have a direction assigned. Directed edges are also known as arcs. Figure B.4



Fig. B.3. The map of Figure B.1 represented as a weighted graph

shows how the directed graph with all the directional properties mentioned above is depicted.



Fig. B.4. A directed graph representing the map of Figure B.1

Sometimes, in a graph, some (but not all) edges are directed. A graph with both directed and undirected edges is called a mixed graph. Obviously, undirected graphs and directed graphs are both special cases of mixed graphs. Mixed graphs, and hence directed graphs, have weighted versions too. Simple graphs in which each pair of nodes is connected by an edge are called complete graphs. Therefore, a complete graph on n (n $\geq $ 2) nodes (denoted by Kn) has$\left(\begin{matrix}n\\2\end{matrix}\right)$edges. A complete graph on five nodes is shown in Figure B.5. In some cases, it is possible to divide the the set of nodes in a graph into two disjoint sets, such that all edges in the graph join a node in one set to a node in the other, and no edge joins nodes in the same set. Such graphs are called bipartite graphs. Simple bipartite graphs, in which all nodes of one set are connected to each node of the other set, are called complete bipartite graphs. If one of the two sets of



Fig. B.5. The complete graph K5

nodes has m nodes in it, and the other set has n nodes, then the complete bipartite graph is denoted by Km;n, (m; n $\geq $1). Figure B.6 depicts the complete bipartite graph K3;2.

Let us consider the problem of traveling between nodes 5 and 9 in the graph of Figure B.2. There are several ways to do it. One may take a direct route from node 5 through nodes 6 and 10 to node 9. Another way could be from node 5 through nodes 6, 7, 4, and 10 to node 9. In graph theory, these routes are called walks. A walk is an alternating sequence of nodes and edges, starting and ending with nodes, where each edge joins the node preceding it with the node succeeding it. In this text we will denote the two walks as 5 – 6 – 10 – 9, and 5 – 6 – 7 – 4 – 5 – 10 – 9, respectively.

A walk in which no two nodes are the same is called a path. So the walk 5 – 6 –10 – 9 is a path, while the walk 5 – 6 – 7 – 4 – 5 – 10 – 9 is not. Walks and paths have equivalents in directed and mixed graphs (directed walk and mixed walk, and directed path and mixed path, respectively). A path in a graph which starts and ends at the same node is called a cycle or tour. In Figure B.2, 5 – 6 – 7 – 4 – 5 denotes a cycle. A graph which does not contain a cycle is called an acyclic graph. The length of a walk, path, or cycle, is the sum of the weights of the edges in it for weighted graphs, and the number of edges in it for unweighted graphs. As in other cases, cycles have equivalents, called directed and mixed cycles, respectively, in directed

and mixed graphs. Walks, paths, cycles, and tours can also be described by the edges or arcs in them. For example, the path 5 – 6 – 10 – 9 can be denoted by the set (5 –6, 6 – 10, 10 – 9).



Fig. B.6. The complete bipartite graph K3;2

**Solution Techniques**

In network optimization (and indeed in many other fields) we need to define systematic procedures to do particular operations. In most cases, we need the procedure to be finite, i.e., if one follows the procedure, one should be able to finish the task in finite time. Such well defined procedures that get over in finite time are called algorithms.

They are called exact if they are guaranteed to output optimal solutions, otherwise they are called approximate. Approximate algorithms are also called heuristics. Now, let us consider the problem of finding a shortest path between two prespecified nodes in a graph. For example, let us suppose that one wants to find a shortest path betweenWarehouse 1 andWarehouse 2 in the map in Figure B.1. If one is willing to look at all paths, compare them and output the cheapest, then one has to consider the following seven options (in terms of nodes in the graph in Figure B.2):

(1) 3 – 2 – 1 – 8 – 9 – 10;

(2) 3 – 2 – 1 – 8 – 5 – 6 – 10;

(3) 3 – 2 – 1 – 8 – 5 – 4 – 7 – 6 – 10;

(4) 3 – 2 – 4 – 7 – 6 – 10;

(5) 3 – 2 – 4 – 7 – 6 – 5 – 8 – 9 – 10;

(6) 3 – 2 – 4 – 5 – 6 – 10; and

(7) 3 – 2 – 4 – 5 – 8 – 9 – 10.

If we generalize this problem to a weighted complete graph with n nodes (n $\geq $ 2), we see that there are 

different paths between any two nodes. (It is left to the reader to check this formula.) This is a daunting number—even if we employed a computer that could enumerate and evaluate a million paths each second, it would take 551.86 years to find a shortest path in a weighted K20 graph.

Algorithms that evaluate all solutions and return the best are called complete enumeration (or exhaustive enumeration) algorithms, and are normally the algorithms of last resort. Typically, there are much smarter ways of solving these problems. For example, there is an algorithm for the shortest path problem due to E.W. Dijkstra which performs an order of n2 computations. In Table B.1 we illustrate the comparison between



and n2 increases as n increase.





Unfortunately, there are network optimization problems for which it is not clear whether such algorithms exist. These are called “hard” (or NP-hard) problems. Examples of these problems are network location problems, like the k-median problem and the facility location problem, and some network routing problems, like the traveling salesperson problem . For hard problems, typically, the solution technique is a refinement of a complete enumeration algorithm. Consequently, in general, we are able to solve easy problems of sizes much larger than that of hard problems. For solving large instances of hard problems, one relies on heuristics, which aim to yield good quality solutions within reasonable times. Unlike exact algorithms that always output optimal solutions, and thus can be compared only on the basis of their execution time and computer memory usage, heuristics can also be compared on the basis of the quality of solutions they output.

**Graph Representations**





Fig. B.7. A simple graph







Note that each column in this matrix has exactly two non-zero elements, a 1 denoting

the head of the corresponding arc, and a -1 denoting the tail of the arc.