MA859: SELECTED TOPICS IN GRAPH THEORY

LECTURE - 20

SHORTEST PATH ALGORITHMS

SHORTEST PATH ALGORITHMS

INTRODUCTION AND EXAMPLES

Suppose that we are given a graph G in which each arc (x, y) has associated with it a number a(x, y) that represents the length of the arc. In some applications, the length may actually represent cost or some other value. The length of a path is defined as the sum of the lengths of the individual arcs comprising the path. For any two vertices, sand t in G, it is possible that there exist several paths from s to t. The shortest-path problem involves finding a path from s to t that has the smallest possible length.

Among all classes of problem in network optimization, shortest-path problems have been some of the most extensively studied. They are commonly encountered in transportation, communication, and production applications and are often imbedded within other types of network optimization problems.

It is not possible here to discuss the many different variations of shortest-path algorithms that exist. Surveys by Deo and Pang (1984), and Gallo and Pallotino (1986), for example, list hundreds of references. Our main focus will be on some of the "classical" algorithms that form the basis for most of the different variations. We will, however, discuss some recent developments. First, let us discuss some applications.

Transportation Planning

Trucks that enter a state must file a route plan with the state government. They are not allowed to travel on roads with prohibitive weight restrictions. On the other hand, the trucker wants to reach his or her destination in the shortest possible time. This problem can be approached by constructing a graph corresponding to the highway network and deleting the arcs on which the truck is prohibited from traveling (or equivalently, giving them a very large length). By knowing the mileage corresponding to each arc as well as the legal speed limit, the travel time on each arc can be computed. Using these times as arc lengths, the trucker would want to solve a shortest-path problem from the point of entry to the final destination.

Salesperson Routing

Suppose that a salesperson in Barmar (Rajasthan) plans to drive to Mysuru (Karnataka) to visit a client. She plans to use the interstate highway system and visit other clients along the route. She knows on the average how much commission she can earn from each proposed stop along the way from Barmar to Mysuru. What route should she take? In this situation, the length of each arc in the graph representing the highway system should be set equal to the net cost (driving less expected commission) of the corresponding highway segment. The salesperson should drive along the shortest path from the Barmar vertex to the Mysuru vertex.
Observe that in this case arc costs will be negative on any arc where the salesperson expects to make a profit and positive on any arc where the salesperson expects to incur a loss. In the previous example, all arc lengths were nonnegative. As we shall see later, these two situations require different solution algorithms.

Investment Planning

A small investor must decide how to invest his funds for the coming year in an optimal fashion. A variety of investments (money market funds, certificates of deposit, savings bonds, etc.) are available. For simplicity, suppose that investments can be made and withdrawn only on the first of each month. Create a vertex corresponding to the first day of each month. Place an arc from vertex x to vertex y for each investment that can be made at time x and mature at time y. Notice that this graph contains no directed cycles. Let the length of each arc equal the negative of the profit earned on the corresponding investment. The best investment plan corresponds to a shortest path (i .e., the path with the most negative total length) from the vertex for time zero to the vertex for one year from now.

Production Lot Sizing

A common problem encountered in production planning is to determine a schedule of production runs when the product's demand varies deterministically over time. We are given a set of demands D1, D2, . . . , Dn over an n-period planning horizon. At each period j, we incur a fixed setup cost Aj if we produce any positive amount as well as a unit production cost Cj. Any amount in excess of the demand for that period is held in inventory until the next period, incurring a per-unit holding cost of Hj.

This problem can be modeled as a mixed integer linear program. Let Q_j be the amount produced in period j and I_j be the amount held over in inventory to period $j+1$. Y_j is a 0-1 variable that is equal to 1 if we produce in period j and equal to 0 otherwise. The model is

$$\text{Min} \sum (C_j Q_j + H_j I_j + A_j Y_j)$$
$$Q_j + I_{j-1} - I_j = D_j \text{ for } j = 1, \dots, n$$
$$Q_j \le M Y_j$$
$$Q_j \ge 0, Y_j = 0, 1$$

It can be shown that an optimal solution will have the property that $Q_j = D_j + D_{j+1} + \dots + D_k$ for some $k \ge j$. That is, production in any period must be a multiple of the demand for some set of future periods. Under these conditions, the problem can be reformulated as a shortest-path problem on a directed graph.

Let W_{jk} be the total cost associated with producing $D_j + D_{j+1} + \dots + D_k$ units in period j.

$$W_{jk} = A_j + C_j Q_j + \sum H_j(I_j + \dots + I_{k-1}) \quad \text{where}$$
$$I_j = D_{j+1} + \dots + D_k,\ I_{j+1} = D_{j+2} + \dots + D_k, \dots, I_{k-1} = D_k$$

The network for a four-period problem is shown in Fig. 1. Any directed path from node 1 to node n represents a sequence of production decisions. The minimum-cost path represents the optimal solution.

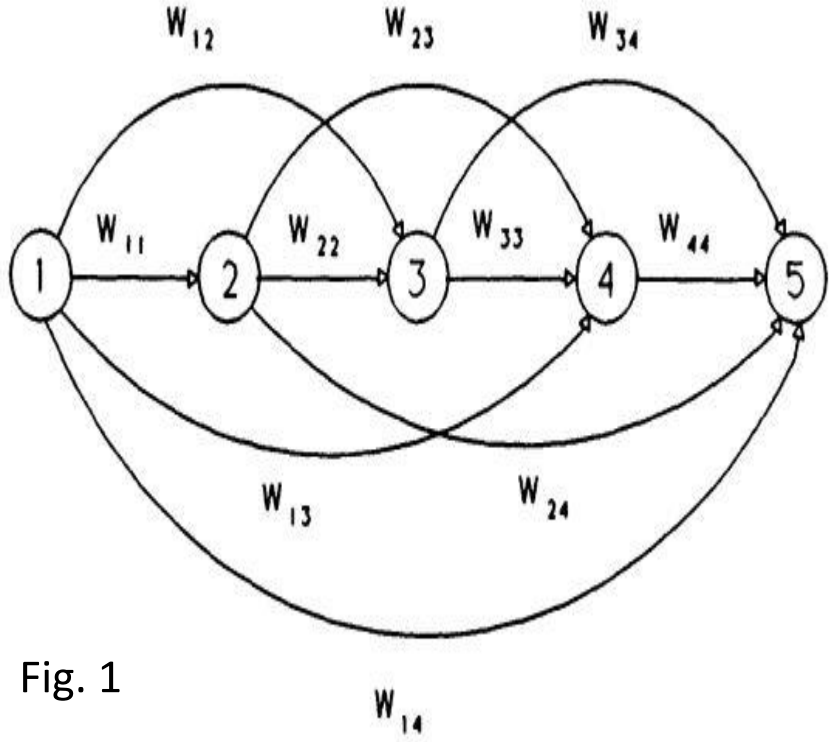
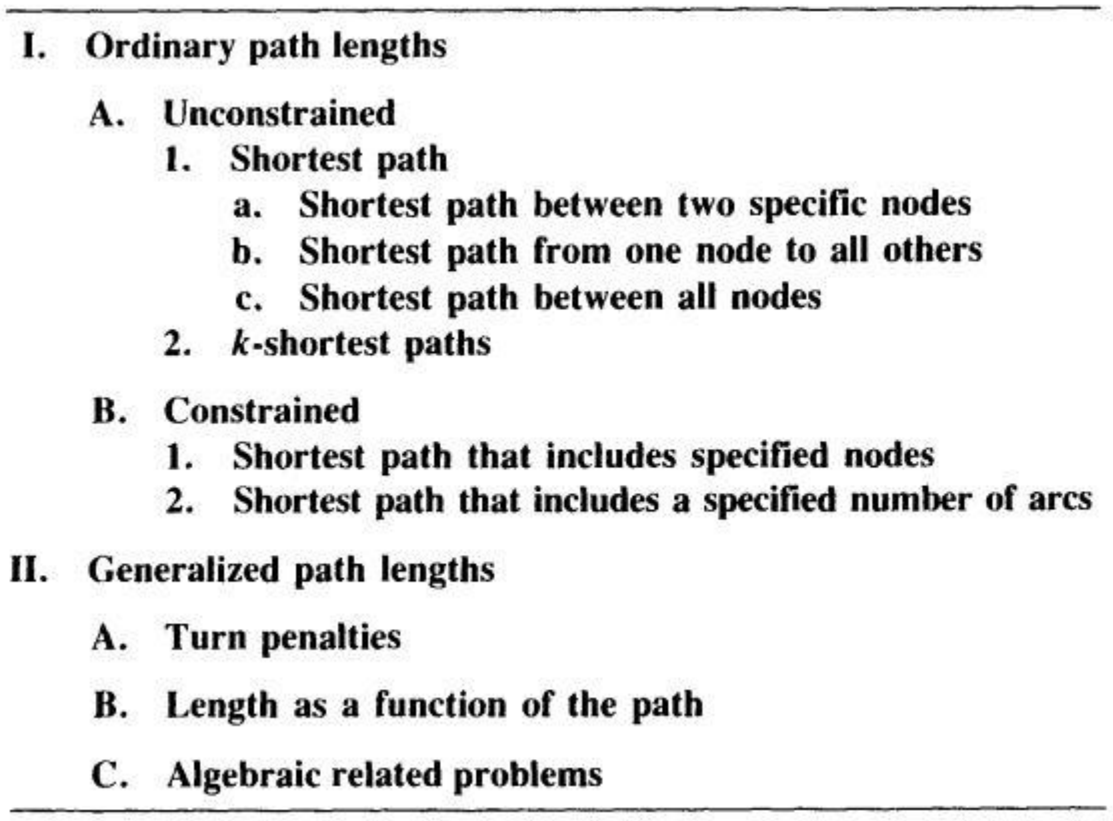


Fig. 1

TYPES OF SHORTEST-PATH PROBLEMS AND ALGORITHMS

We can classify shortest-path problems into many different types. Following is the summary of this classification:



I. **Ordinary path lengths**
   - A. **Unconstrained**
     - 1. **Shortest path**
       - a. **Shortest path between two specific nodes**
       - b. **Shortest path from one node to all others**
       - c. **Shortest path between all nodes**
     - 2. ***k*-shortest paths**
   - B. **Constrained**
     - 1. **Shortest path that includes specified nodes**
     - 2. **Shortest path that includes a specified number of arcs**

II. **Generalized path lengths**
   - A. **Turn penalties**
   - B. **Length as a function of the path**
   - C. **Algebraic related problems**

By ordinary path length, we mean that the path length is the sum of the lengths of the individual arcs. Generalized path length refers to problems in which the length is a more complicated function of arc lengths.

For example, in transportation networks, one usually includes a “turn penalty” for making a turn at an intersection. Other real-valued functions might be used to determine the path length. Here we shall be concerned with problems in category I. A.1: unconstrained problems with ordinary path lengths involving finding shortest paths between nodes or the k-shortest paths (i.e., the first-shortest, second-shortest, and so on) in a network. Constrained problems involve specifying intermediate nodes or a certain number of arcs that must be visited along the path. The reader is referred to the survey by Deo and Pang (1984) for a more detailed classification of shortest-path problems.

Two general algorithmic approaches are used to solve shortest-path problems. Label-setting methods apply to networks with nonnegative arc lengths. Label-correcting methods apply to networks with negative arc lengths as well. Both approaches assign tentative labels to nodes at each step. These labels represent the shortest path distances that are known at that point of the algorithm. Label-setting methods set one or more permanent labels at each iteration. Permanent labels represent the actual shortest distances in the optimal solution. In label-correcting methods, all labels are temporary until the final iteration, at which time they all become permanent. As we shall see, algorithms for ordinary shortest-path problems have polynomial time complexity.

SHORTEST PATHS FROM A SINGLE SOURCE

We shall consider algorithms that generate a path from node s to node t that has the smallest possible length. With simple modifications, they can also find the shortest paths from node s to all other nodes.

Dijkstra's Shortest-Path Algorithm

The following algorithm is due to Dijkstra (1959) and provides the basis for the most efficient algorithms known for solving this problem. Most computational improvements for solving shortest-path problems have resulted from improving the data structures used to implement Dijkstra's algorithm. Dijkstra's algorithm is a label-setting algorithm in that a label is made permanent at each iteration.

The main idea underlying the Dijkstra shortest-path algorithm is quite simple. Suppose we know the k vertices that are closest in total length to vertex s in the graph and also a shortest path from s to each of these vertices. (Of course, a shortest path from vertex s to itself is the null path consisting of no arcs, which has length equal to zero.) Label vertex s and these k vertices with their shortest distance from s. Then the (k + 1)th closest vertex to x is found as follows. For each unlabeled vertex y, construct k distinct paths from s to y by joining the shortest path from s to x with arc (x, y) for all labeled vertices x (see Fig. 2). Select the shortest of these k paths and let it tentatively be the shortest path from s to y.

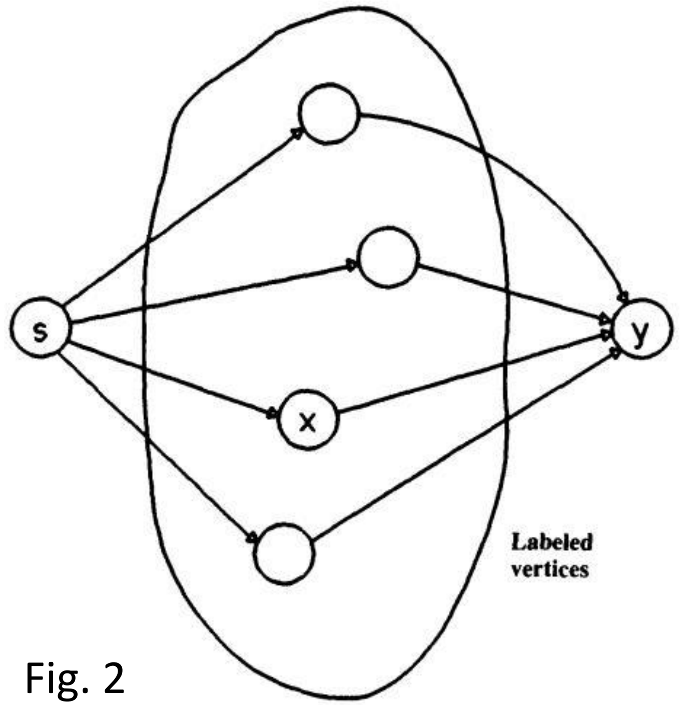


Fig. 2

Which labeled vertex is the (k + 1)th closest vertex to s? It is the unlabeled vertex with the shortest tentative path from s as calculated above. This follows because the shortest path from s to the (k + 1)th closest vertex to s must use only labeled vertices as its intermediate vertices since all arc lengths are nonnegative.
Therefore, if the k closest vertices to s are known, the (k + 1)st can be determined as above. Starting with k = 0, this process can be repeated until the shortest path from s to t has been found (that is, t is labeled). With this in mind as motivation, we can now formally state the Dijkstra shortest-path algorithm.

Step 1. Initially, all arcs and vertices are unlabeled. Assign a number d(x) to each vertex x to denote the tentative length of the shortest path from s to x that uses only labeled vertices as intermediate vertices. Initially, set d(s) = 0 and d(x) = ∞ for all x =1= s. Let y denote the last vertex that was labeled. Label vertex s and let y = s.

Step 2. For each unlabeled vertex x, redefine d(x) as follows:

$$d(x) = \min\{d(x), d(y) + a(y, x)\} \qquad (1)$$

This can be performed efficiently by scanning the forward star of node y since only these nodes will be affected. If d(x) = ∞ for all unlabeled vertices x, then stop because no path exists from s to any unlabeled vertex. Otherwise, label the unlabeled vertex x with the smallest value of d(x). Also label the arc directed into vertex x from a labeled vertex that determined the value of d(x) in the above minimization. Let y = x.

Step 3. If vertex t has been labeled then stop, since a shortest path from s to t has been discovered. This path consists of the unique path of labeled arcs from s to t. If vertex t has not been labeled yet, repeat step 2.

Note that whenever the algorithm labels a vertex (except vertex s) the algorithm also labels an arc directed into this vertex. Thus, each vertex has at most one labeled arc directed into it, and the labeled arcs cannot contain a cycle since no arc is labeled if both its endpoints have a labeled arc incident to it. Therefore, we can conclude that the labeled arcs form an arborescence rooted at s. This arborescence is called a shortest-path arborescence. The unique path from s to any other vertex x contained in any shortest-path arborescence is a shortest path from s to x.

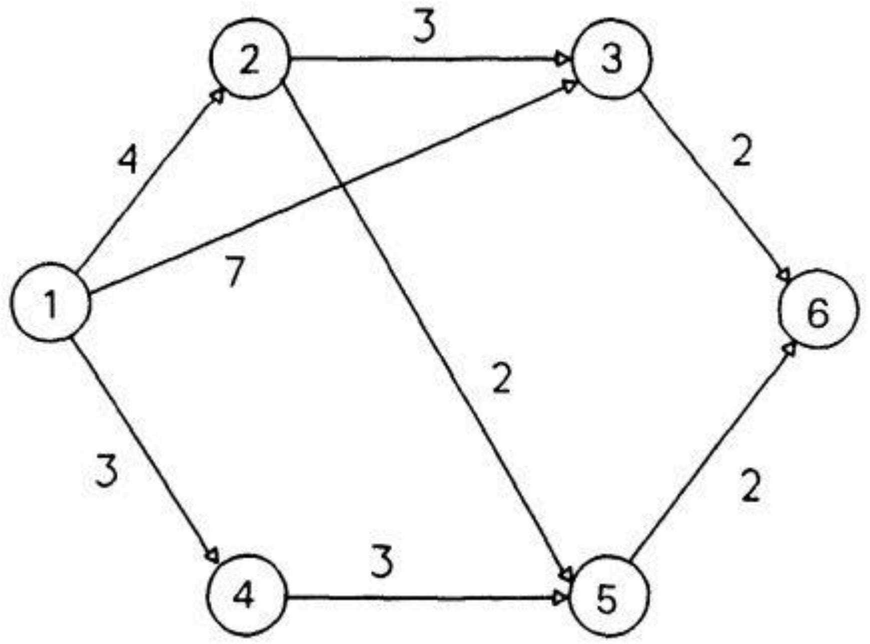
If the shortest path from s to x in a shortest -path arborescence passes through vertex y, it follows that the portion of this path from y to x is a shortest path from y to x. Otherwise, there exists another, even shorter, path from y to x, which contradicts that we found a shortest path from s to x.

Since the labeled arcs at all times form an arborescence, the algorithm can be regarded as the growing of an arborescence rooted at vertex s. Once vertex t is reached, the growing process can be terminated.

If you want to determine a shortest path from vertex s to every other vertex in the graph, the growing process could be continued until all vertices were included in the shortest-path arborescence, in which case the arborescence would become a spanning arborescence (if one exists). In this case, step 3 would read as follows:

Step 3. If all vertices have been labeled, stop because the unique path of labeled arcs from s to x is a shortest path from s to x for all vertices x. Otherwise, return to step 2.

EXAMPLE 1: Perform the Dijkstra shortest-path algorithm to find a shortest path from node 1 to node 6 in the graph in the following Fig.



Step 1: Initially, only node 1 is permanently labeled and d(1)=0. Assign tentative distances d(x) = ∞ for all x ≠1.

Step 2: Recompute tentative distances for the remaining nodes:

d(2)=min{d(2), d(1)+a(1,2)=min{∞,0+4}=4

d(3)=min{d(3), d(1)+a(1,3)=min{∞,0+7}=7

d(4)=min{d(4), d(1)+a(1,4)=min{∞,0+3}=3

Since the minimum distance is on node 4, d(4)=3; we label node 4 permanently and arc (1,4). The current shortest path consists of arc(1,4).

We now look for the shortest distance from node 4 (which has been permanently labeled).

No change in d(2) and d(3) as they are not adjancent to node 4.

d(5)=min{d(5), d(4)+a(4,5)}=min{∞, 3+3}=6

Now the minimum distance is on node 2, d(2)=4 (permanently label it).

d(3)=min{d(3), d(2)+a(2,3)}= min{7, 4+3}=7

d(5)=min{d(5), d(2)+a(2,5)}=min{6, 4+2}=6

Since the minimum distance is on node 5, d(5)=6 (permanently label it).

Now d(6)=min{d(6), d(5)+a(5,6)}=min{∞,6+2}=8

Hence the shortest path from 1 to 6 is 1→4→5→6 and the shortest distance is 8.

// End of Lecture //