Lecture 18
Solving Shortest Path Problem: Dijkstra’s Algorithm

October 23, 2009
Outline

• Focus on Dijkstra’s Algorithm
  • Importance: Where it has been used?
  • Algorithm’s general description
  • Algorithm steps in detail
  • Example
One-To-All Shortest Path Problem

We are given a weighted network \((V, E, C)\) with node set \(V\), edge set \(E\), and the weight set \(C\) specifying weights \(c_{ij}\) for the edges \((i, j) \in E\). We are also given a starting node \(s \in V\). The one-to-all shortest path problem is the problem of determining the shortest path from node \(s\) to all the other nodes in the network.

The weights on the links are also referred as costs.
Algorithms Solving the Problem

- Dijkstra’s algorithm
  - Solves only the problems with nonnegative costs, i.e.,
    \[ c_{ij} \geq 0 \text{ for all } (i, j) \in E \]

- Bellman-Ford algorithm
  - Applicable to problems with arbitrary costs

- Floyd-Warshall algorithm
  - Applicable to problems with arbitrary costs
  - Solves a more general all-to-all shortest path problem

Floyd-Warshall and Bellman-Ford algorithm solve the problems on graphs that do not have a cycle with negative cost.
Importance of Dijkstra's algorithm

Many more problems than you might at first think can be cast as shortest path problems, making Dijkstra’s algorithm a powerful and general tool. For example:

• Dijkstra’s algorithm is applied to automatically find directions between physical locations, such as driving directions on websites like Mapquest or Google Maps.

• In a networking or telecommunication applications, Dijkstra’s algorithm has been used for solving the min-delay path problem (which is the shortest path problem). For example in data network routing, the goal is to find the path for data packets to go through a switching network with minimal delay.

• It is also used for solving a variety of shortest path problems arising in plant and facility layout, robotics, transportation, and VLSI* design

*Very Large Scale Integration
General Description

Suppose we want to find a shortest path from a given node $s$ to other nodes in a network (one-to-all shortest path problem)

- Dijkstra’s algorithm solves such a problem
  - It finds the shortest path from a given node $s$ to all other nodes in the network
  - Node $s$ is called a starting node or an initial node

- How is the algorithm achieving this?
  - Dijkstra’s algorithm starts by assigning some initial values for the distances from node $s$ and to every other node in the network
  - It operates in steps, where at each step the algorithm improves the distance values.
  - At each step, the shortest distance from node $s$ to another node is determined
The algorithm characterizes each node by its state. The state of a node consists of two features:

- **Distance value** and **status label**

  - Distance value of a node is a scalar representing an estimate of the its distance from node $s$.
  
  - Status label is an attribute specifying whether the distance value of a node is equal to the shortest distance to node $s$ or not.

  - The status label of a node is **Permanent** if its distance value is equal to the shortest distance from node $s$.
  
  - Otherwise, the status label of a node is **Temporary**

The algorithm maintains and step-by-step updates the states of the nodes. At each step one node is designated as **current**.
**Notation**

In what follows:

- $d_\ell$ denotes the distance value of a node $\ell$.

- $p$ or $t$ denotes the status label of a node, where $p$ stand for permanent and $t$ stands for temporary.

- $c_{ij}$ is the cost of traversing link $(i, j)$ as given by the problem.

The state of a node $\ell$ is the ordered pair of its distance value $d_\ell$ and its status label.
Algorithm Steps

Step 1. Initialization

- Assign the zero distance value to node $s$, and label it as **Permanent**. [The state of node $s$ is $(0, p)$.]
- Assign to every node a distance value of $\infty$ and label them as **Temporary**. [The state of every other node is $(\infty, t)$.]
- Designate the node $s$ as the **current** node
Step 2. Distance Value Update and Current Node Designation Update

Let $i$ be the index of the current node.

1. Find the set $J$ of nodes with temporary labels that can be reached from the current node $i$ by a link $(i, j)$. Update the distance values of these nodes.

   - For each $j \in J$, the distance value $d_j$ of node $j$ is updated as follows
     \[
     \text{new } d_j = \min\{d_j, d_i + c_{ij}\}
     \]
     where $c_{ij}$ is the cost of link $(i, j)$, as given in the network problem.

2. Determine a node $j$ that has the smallest distance value $d_j$ among all nodes $j \in J$,

   \[
   \text{find } j^* \text{ such that } \min_{j \in J} d_j = d_j^*
   \]

3. Change the label of node $j^*$ to permanent and designate this node as the current node.
**Step 3.** *Termination Criterion*

If all nodes that can be reached from node $s$ have been permanently labeled, then stop - we are done.

If we cannot reach any temporary labeled node from the current node, then all the temporary labels become permanent - we are done.

Otherwise, go to Step 2.
We want to find the shortest path from node 1 to all other nodes using Dijkstra’s algorithm.
Initialization - Step 1

- Node 1 is designated as the current node
- The state of node 1 is \((0,p)\)
- Every other node has state \((\infty,t)\)
Step 2

- Nodes 2, 3, and 6 can be reached from the current node 1.
- Update distance values for these nodes:
  \[ d_2 = \min\{\infty, 0 + 7\} = 7 \]
  \[ d_3 = \min\{\infty, 0 + 9\} = 9 \]
  \[ d_6 = \min\{\infty, 0 + 14\} = 14 \]
- Now, among the nodes 2, 3, and 6, node 2 has the smallest distance value.
- The status label of node 2 changes to permanent, so its state is \((7, p)\), while the status of 3 and 6 remains temporary.
- Node 2 becomes the current node.
Step 3

Graph at the end of Step 2

We are not done, not all nodes have been reached from node 1, so we perform another iteration (back to Step 2)
Another Implementation of Step 2

- Nodes 3 and 4 can be reached from the current node 2.
- Update distance values for these nodes:

\[ d_3 = \min\{9, 7 + 10\} = 9 \]
\[ d_6 = \min\{\infty, 7 + 15\} = 22 \]

- Now, between the nodes 3 and 4 node 3 has the smallest distance value.
- The status label of node 3 changes to permanent, while the status of 6 remains temporary.
- Node 3 becomes the current node.

We are not done (Step 3 fails), so we perform another Step 2.
Another Step 2

- Nodes 6 and 4 can be reached from the current node 3
- Update distance values for them

\[ d_4 = \min\{22, 9 + 11\} = 20 \]
\[ d_6 = \min\{14, 9 + 2\} = 11 \]

- Now, between the nodes 6 and 4 node 6 has the smallest distance value
- The status label of node 6 changes to permanent, while the status of 4 remains temporary
- Node 6 becomes the current node

We are not done (Step 3 fails), so we perform another Step 2
Another Step 2

- Node 5 can be reached from the current node 6
- Update distance value for node 5

\[ d_5 = \min\{\infty, 11 + 9\} = 20 \]

- Now, node 5 is the only candidate, so its status changes to permanent
- Node 5 becomes the current node

From node 5 we cannot reach any other node. Hence, node 4 gets permanently labeled and we are done.
Chapter 6.3.2 in your book has another example of the implementation of Dijkstra’s algorithm