Bang Ye Wu and Kun-Mao Chao

Spanning Trees and Optimization Problems (Excerpt)

CRC PRESS
Boca Raton London New York Washington, D.C.

Chapter 3

Shortest-Paths Trees

Consider a connected, undirected network with one special node, called the source (or root). Associated with each edge is a distance, a nonnegative number. The objective is to find the set of edges connecting all nodes such that the sum of the edge lengths from the source to each node is minimized. We call it a *shortest-paths tree* (SPT) rooted at the source.

Shortest-paths trees are not necessarily unique. Figure 3.1 gives two shortest-paths trees rooted at vertex a.

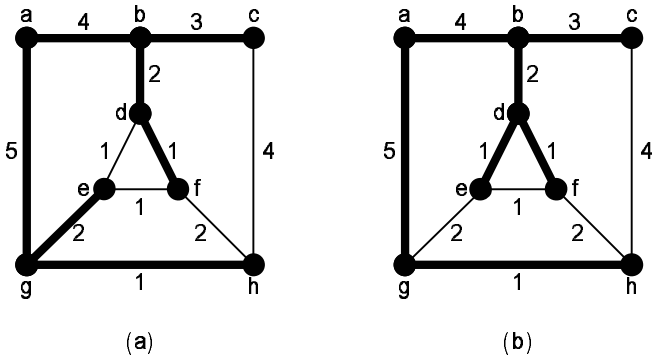

FIGURE 3.1: A shortest-paths tree rooted at vertex a.

Consider the graph in Figure 3.2.

In this chapter, the terms "edge" and "arc" are used interchangeably. We discuss two well-known algorithms for constructing a shortest-paths tree: Dijkstra's algorithm and the Bellman-Ford algorithm. Dijkstra's algorithm assumes that all edge weights in the graph are nonnegative, whereas the Bellman-Ford algorithm allows negative-weight edges in the graph. If there is no negative-weight cycle, the Bellman-Ford algorithm produces the shortest paths and their weights. Otherwise, the algorithm detects the negative cycles and indicates that no solution exists.

Algorithm: DIJKSTRA

Input: A weighted, directed graph $G = (V, E, w)$; a source vertex s.
Output: A shortest-paths spanning tree T rooted at s.

```
for each vertex v ∈ V do
    δ[v] ← ∞
    π[v] ←NIL
δ[s] ← 0
T ← ∅
S ← V
while S ≠ ∅ do
    Choose u ∈ S with minimum δ[u]
    S ← S − {u}
    if u ≠ s then T ← T ∪ {(π[u], u)}
    for each vertex v adjacent to u do
        if δ[v] > δ[u] + w(u, v) then
            δ[v] ← δ[u] + w(u, v)
            π[v] ← u
```

Consider the graph in Figure 3.3.
Figure 3.4.
Figure 3.5.
Figure 3.6.
Figure 3.7.
Figure 3.8.
Figure 3.9.
Figure 3.10.
Figure 3.11.
Figure 3.12 gives the shortest-paths tree constructed by Dijkstra's algorithm.

Figure 3.13 gives an example of a directed graph with negative-weight edges for which Dijkstra's algorithm produces an incorrect answer.

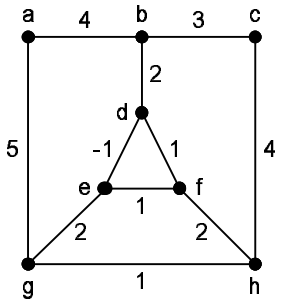

FIGURE 3.2: An undirected graph with a negative-weight edge.

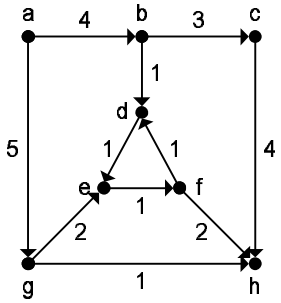

FIGURE 3.3: A weighted, directed graph.

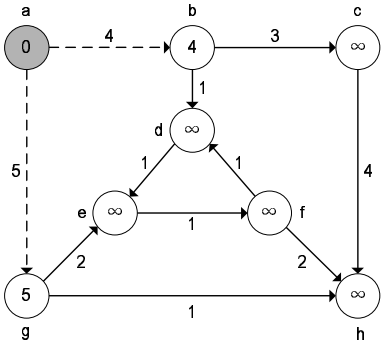

FIGURE 3.4: Vertex a is chosen, and edges (a, b) and (a, g) are relaxed.

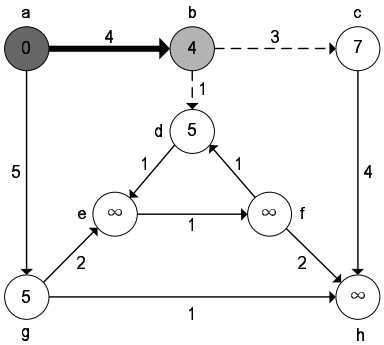

FIGURE 3.5: Vertex b is chosen, and edges (b, c) and (b, d) are relaxed. Edge (a, b) is added to the shortest-paths tree.

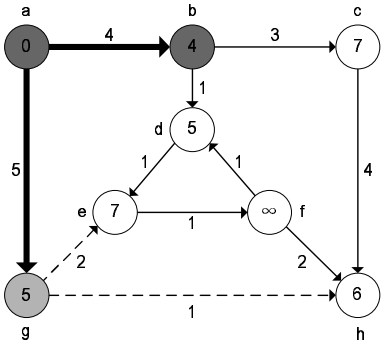

FIGURE 3.6: Vertex g is chosen, and edges (g, e) and (g, h) are relaxed. Edge (a, g) is added to the shortest-paths tree.

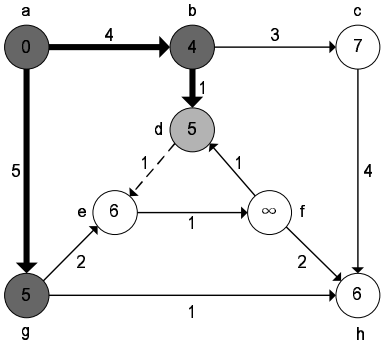

FIGURE 3.7: Vertex d is chosen, and edge (d, e) is relaxed. Edge (b, d) is added to the shortest-paths tree.

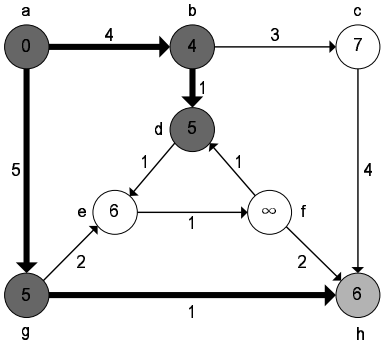

FIGURE 3.8: Vertex h is chosen. Edge (g, h) is added to the shortest-paths tree.

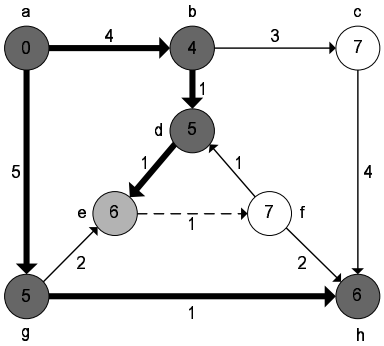

FIGURE 3.9: Vertex e is chosen, and edge (e, f) is relaxed. Edge (d, e) is added to the shortest-paths tree.

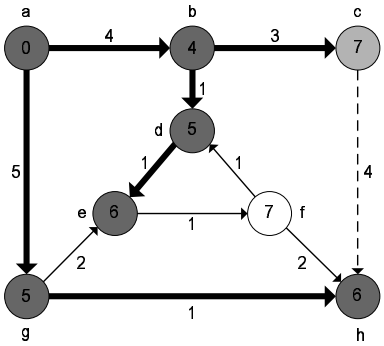

FIGURE 3.10: Vertex c is chosen, and edge (c, h) is relaxed. Edge (b, c) is added to the shortest-paths tree.

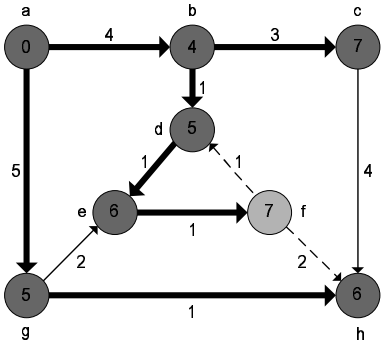

FIGURE 3.11: Vertex f is chosen, and edges (f, d) and (f, h) are relaxed. Edge (e, f) is added to the shortest-paths tree.

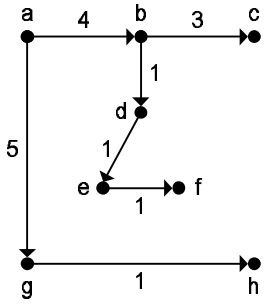

FIGURE 3.12: A shortest-paths tree constructed by Dijkstra's algorithm for the graph in Figure 3.3.

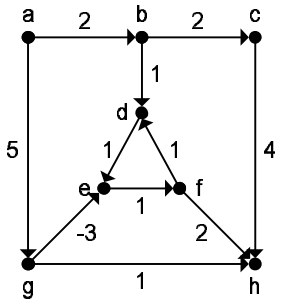

FIGURE 3.13: A weighted, directed graph with a negative-weight edge.

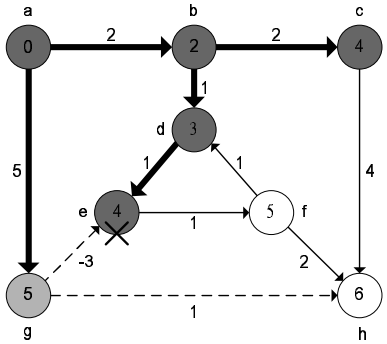

FIGURE 3.14: A failure of Dijkstra's algorithm.

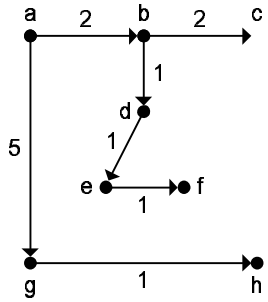

FIGURE 3.15: A wrong shortest-paths tree.

As Dijkstra's algorithm proceeds on the graph in Figure 3.13, it will reach the status shown in Figure 3.14.

The graph in Figure 3.15 is a wrong shortest-path tree produced by Dijkstra's algorithm.

A correct shortest-paths tree is given in Figure 3.16.

Algorithm: BELLMAN-FORD
Input: A weighted, directed graph $G = (V, E, w)$; a source vertex s.
Output: A shortest-paths spanning tree T rooted at s.

```
for each vertex v ∈ V do
    δ[v] ← ∞
    π[v] ←NIL
δ[s] ← 0
for i ← 0 to n − 1 do
    for each (u, v) ∈ E do
        if δ[v] > δ[u] + w(u, v) then
            δ[v] ← δ[u] + w(u, v)
```

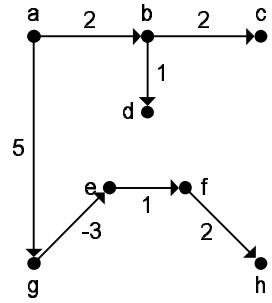

FIGURE 3.16: A correct shortest-paths tree.

$\pi[v] \leftarrow u$
for each $(u, v) \in E$ **do**
  if $\delta[v] > \delta[u] + w(u, v)$ **then**
    Output "A negative cycle exists."
    Exit
$T \leftarrow \emptyset$
for $v \in V - s$ **do**
  $T \leftarrow T \cup \{(\pi[v], v)\}$

Figure 3.17.
Figure 3.18.
Figure 3.19.
Figure 3.20.
Figure 3.21.

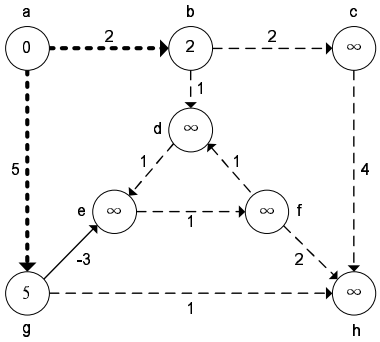

FIGURE 3.17: Shortest-path estimates $\delta[b]$ and $\delta[g]$ are modified.

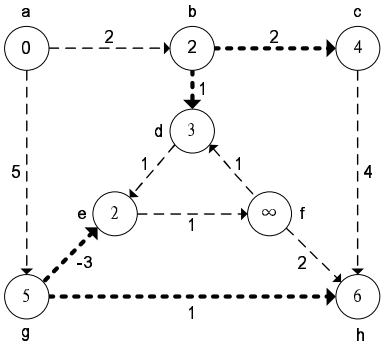

FIGURE 3.18: Shortest-path estimates $\delta[c]$, $\delta[d]$, $\delta[e]$, and $\delta[h]$ are modified.

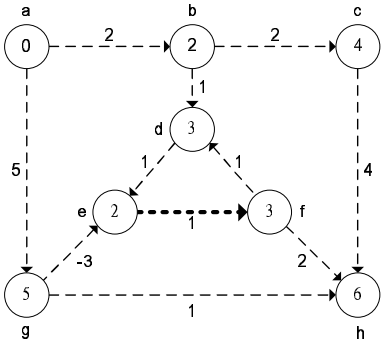

FIGURE 3.19: Shortest-path estimate $\delta[f]$ is modified.

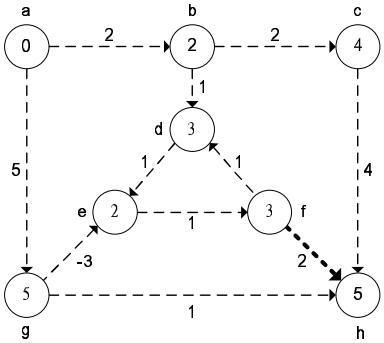

FIGURE 3.20: Shortest-path estimate $\delta[h]$ is modified.

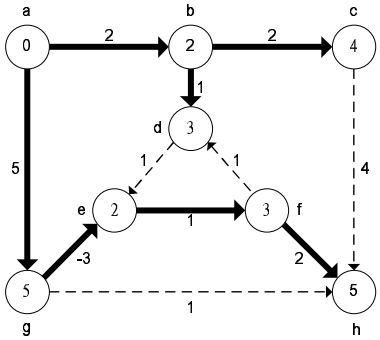

FIGURE 3.21: The shortest-paths tree constructed by the Bellman-Ford algorithm.

Bibliographic Notes and Further Reading

The shortest-paths tree problem is one of the most classical network flow optimization problems. An equivalent problem is to find a shortest path from a given source vertex $s \in V$ to every vertex $v \in V$. Algorithms for this problem have been studied for a long time. In fact, since the end of the 1950s, thousands of scientific works have been published. A good description of the classical algorithms and their implementations can be found in [2, 8].

Dijkstra's original algorithm, by Edsger W. Dijkstra [5], did not mention the usage of a priority queue. A discussion of using different priority queue techniques can be found in [2, 3].

For the shortest path problem with nonnegative arc lengths, the Fibonacci heap data structure [7] yields an $O(m + n \log n)$ implementation of Dijkstra's algorithm in the pointer model of computation. Let U denote the biggest arc length, and C be the ratio between U and the smallest nonzero arc length. In a RAM model with word operations, the fastest known algorithms achieve the following bounds: $O(m + n(\sqrt{\log n})$ [12], $O(m + n(\log C \log\log C)^{1/3})$ [9, 13], $O(m \log\log U)$ [10], and $O(m \log\log n)$ [15]. Ulrich Meyer [11] shows the problem can be solved in linear average time if input arc lengths are independent and uniformly distributed. Andrew V. Goldberg [9] shows that a simple modification of the algorithm of [4] yields an algorithm with linear average running time on the uniform arc length distribution. For undirected graphs, Mikkel Thorup [14] gave a linear-time algorithm in a word RAM model.

The Bellman-Ford algorithm is based on separate algorithms by Richard Bellman [1], and Lestor Ford, Jr. and D.R. Fulkerson [6]. Though the Bellman-Ford algorithm is simple and has a high running time, to date there

is no algorithm which significantly improves its asymptotic complexity.

References

[1] R. Bellman. On a routing problem. *Quar. Appl. Math.*, 16:87–90, 1958.

[2] B.V. Cherkassky, A.V. Goldberg, and T. Radzik. Shortest paths algorithms: theory and experimental evaluation. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 516–525, 1994.

[3] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. The MIT Press, 1994.

[4] E.V. Denardo and B.L. Fox. Shortest-route methods: 1. reaching, pruning, and buckets. *Oper. Res.*, 27:161–186, 1979.

[5] E.W. Dijkstra. A note on two problems in connection with graphs. *Numer. Math.*, 1:269–271, 1959.

[6] L.R. Ford, Jr. and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[7] M.L. Fredman and R.E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34:596–615, 1987.

[8] G. Gallo and S. Pallottino. Shortest paths algorithms. *Ann. Oper. Res.*, 13:3–79.

[9] A.V. Goldberg. A simple shortest path algorithm with linear average time. In *Proceedings of the 9th Annual European Symposium Algorithms*, pages 230–241, 2001.

[10] T. Hagerup. Improved shortest paths in the word RAM. In *Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 61–72, 2000.

[11] U. Meyer. Single-source shortest paths on arbitrary directed graphs inlinear average time. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 797–806, 2001.

[12] R. Raman. Priority queues: small, monotone and trans-dichotomous. In *Proceedings of the 4th Annual European Symposium Algorithms*, pages 121–137, 1996.

[13] R. Raman. Recent results on single-source shortest paths problem. *SIGACT News*, 28:81–87, 1997.

[14] M. Thorup. Undirected single-source shortest paths with positive integer weights in linear time. *J. ACM*, 46:362–394, 1999.

[15] M. Thorup. On RAM priority queues. *SIAM J. Comput.*, 30:86–109, 2000.