Bang Ye Wu and Kun-Mao Chao

Spanning Trees and Optimization Problems (Excerpt)

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

Chapter 2

Minimum Spanning Trees

What is a minimum spanning tree for the weighted graph in Figure 2.1? Notice that a minimum spanning tree is not necessarily unique.

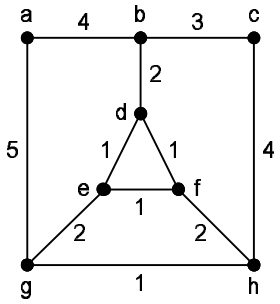

FIGURE 2.1: A weighted graph.

Figure 2.2 gives four minimum spanning trees, where each of them is of total weight 14.

Let $G + e$ denote the graph obtained by inserting edge e into G.

LEMMA 2.1

Any two vertices in a tree are connected by a unique path.

LEMMA 2.2

Let T be a spanning tree of a graph G, and let e be an edge of G not in T. Then $T + e$ contains a unique cycle.

THEOREM 2.1

Let $F_1, F_2, \ldots, F_k$ be a spanning forest of G, and let (u, v) be the smallest of all edges with only one endpoint $u \in V(F_1)$. Then there is an optimal one containing (u, v) among all spanning trees containing all edges in $\cup_{i=1}^{k} E(F_i)$.

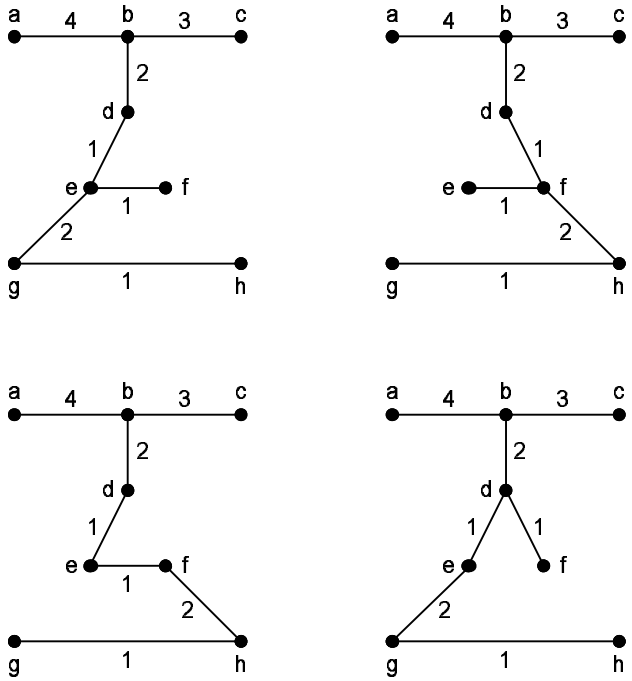

FIGURE 2.2: Some minimum spanning trees.

Algorithm: BORŮVKA
Input: A weighted, undirected graph $G = (V, E, w)$.
Output: A minimum spanning tree T

```
T ← ∅
while |T| < n − 1 do
    F ← a forest consisting of the smallest edge incident to
        each vertex in G
    G ← G\F
    T ← T ∪ F
```

Figure 2.3 illustrates the execution of the BORŮVKA algorithm on the graph from Figure 2.1.

Algorithm: PRIM
Input: A weighted, undirected graph $G = (V, E, w)$.
Output: A minimum spanning tree T.

```
T ← ∅
```

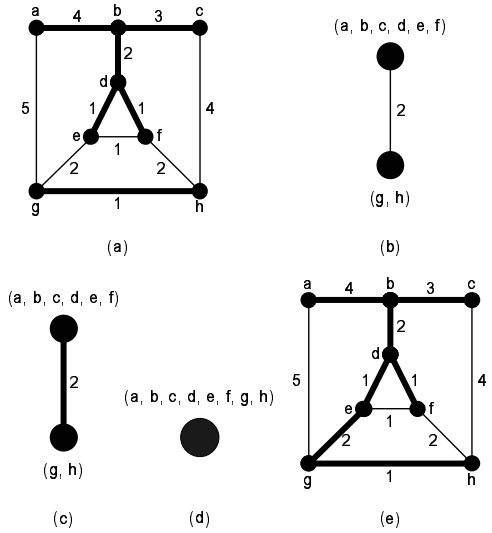

FIGURE 2.3: The execution of the BORŮVKA algorithm on the graph from Figure 2.1.

Let r be an arbitrarily chosen vertex from V.
$U \leftarrow \{r\}$
while $|U| < n$ **do**
    Find $u \in U$ and $v \in V - U$ such that the edge (u, v) is a smallest
        edge between U and $V - U$.
    $T \leftarrow T \cup \{(u, v)\}$
    $U \leftarrow U \cup \{v\}$

Figure 2.4 illustrates the execution of the PRIM algorithm.

Algorithm: KRUSKAL
Input: A weighted, undirected graph $G = (V, E, w)$.
Output: A minimum spanning tree T.
    Sort the edges in E in nondecreasing order by weight.
    $T \leftarrow \emptyset$
    Create one set for each vertex.
    for each edge (u, v) in sorted order **do**
        $x \leftarrow$ FIND(u)
        $y \leftarrow$ FIND(v)
        if $x \neq y$ **then**
            $T \leftarrow T \cup \{(u, v)\}$
            UNION(x, y)

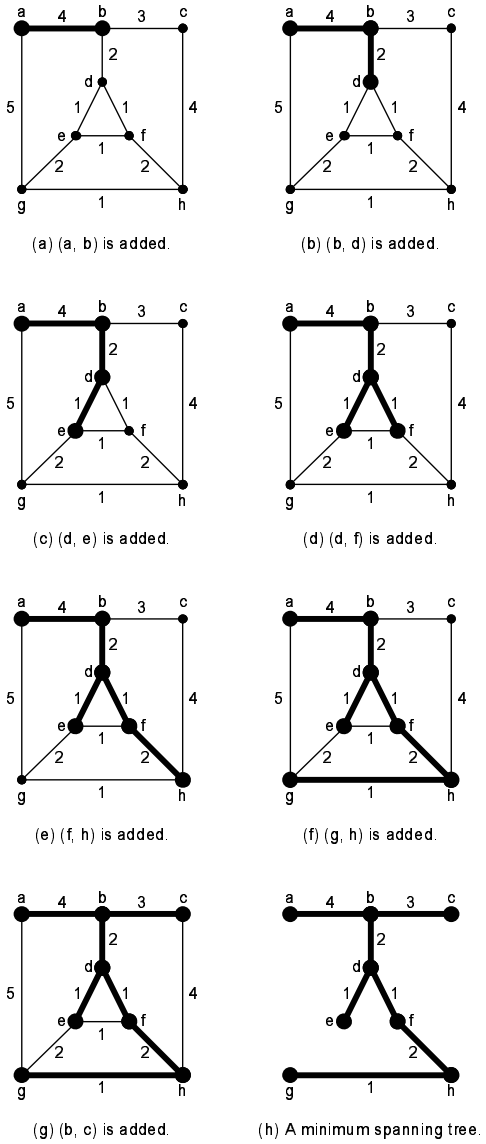

FIGURE 2.4: The execution of the PRIM algorithm on the graph from Figure 2.1.

Figure 2.5 illustrates the execution of the KRUSKAL algorithm.

Bibliographic Notes and Further Reading

The history of the *minimum spanning tree* (MST) problem is long and rich. An excellent survey paper by Ronald Graham and Pavol Hell [9] describes the history of the problem up to 1985. The earliest known MST algorithm was proposed by Otakar Borůvka [1], a great Czech mathematician, in 1926. At that time, he was considering an efficient electrical coverage of Bohemia, which occupies the western and middle thirds of today's Czech Republic. In the mid-1950s when the computer age just began, the MST problem was attacked again by several researchers. Among them, Joseph Kruskal [14] and Robert Prim [16] gave two commonly used textbook algorithms. Both of them mentioned Borůvka's paper. In fact, Prim's algorithm was a rediscovery of the algorithm by the prominent number theoretician Vojtěch Jarník [10].

Textbook algorithms run in $O(m \log n)$ time. Andrew Chi-Chih Yao [19], and David Cheriton and Robert Tarjan [4] independently made improvements to $O(m \log \log n)$. By the invention of Fibonacci heaps, Michael Fredman and Robert Tarjan [7] reduced the complexity to $O(m\beta(m, n))$, where $\beta(m, n) = \min\{i | \log^i n \leq m/n\}$. In the worst case, $m = O(n)$ and the running time is $O(m \log^* m)$. The complexity was further lowered to $O(m \log \beta(m, n))$ by Harold N. Gabow, Zvi Galil, Thomas H. Spencer, and Robert Tarjan [8].

On the other hand, David Karger, Philip Klein, and Robert Tarjan [11] gave a randomized linear-time algorithm to find a minimum spanning tree in the restricted random-access model. If the edge costs are integer and the models allow bucketing and bit manipulation, Michael Fredman and Dan Willard [6] gave a deterministic linear-time algorithm.

Given a spanning tree, how fast can we verify that it is minimum? Robert Tarjan [18] gave an almost linear-time algorithm by using path compression. János Komlós [13] showed that a minimum spanning tree can be verified in linear number of comparisons, but with nonlinear overhead to decide which comparisons to make. Brandon Dixon, Monika Rauch, and Robert Tarjan [5] gave the first linear-time verification algorithm. Valerie King [12] proposed a simpler linear-time verification algorithm. All these methods use the fact that a spanning tree is a minimum spanning tree if and only if the weight of each nontree edge (u, v) is at least the weight of the heaviest edge in the path in the tree between u and v.

It remains an open problem whether a linear-time algorithm exists for finding a minimum spanning tree. Bernard Chazelle [2] took a significant step towards a solution and charted out a new line of attack. His algorithm runs

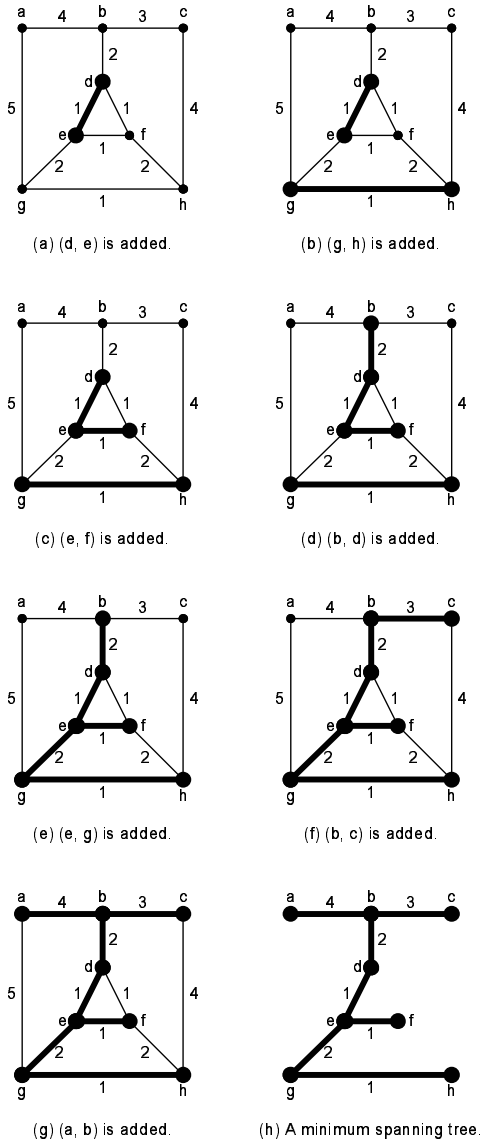

FIGURE 2.5: The execution of the KRUSKAL algorithm on the graph from Figure 2.1.

in $O(m\alpha(m,n))$ time, where α is the functional inverse of Ackermann's function defined in [17]. The key idea is to compute suboptimal independent sets in a nongreedy fashion, and then progressively improve upon them until an optimal solution is reached. An approximate priority queue, called a *soft heap* [3], is used to construct a suboptimal spanning tree, whose quality is progressively refined until a minimum spanning tree is finally produced.

Seth Pettie and Vijaya Ramachandran [15] established that the algorithmic complexity of the minimum spanning tree problem is equal to its decision-tree complexity. They gave a deterministic, comparison-based MST algorithm that runs in $O(T^*(m,n))$, where $T^*(m,n)$ is the number of edge-weight comparisons needed to determine the MST. Because of the nature of their algorithm, its exact running time is unknown. The source of their algorithm's mysterious running time, and its optimality, is the use of precomputed "MST decision trees" whose exact depth is unknown but nonetheless provably optimal. A trivial lower bound is $\Omega(m)$; and the best upper bound is $O(m\alpha(m,n))$ [2].

References

[1] O. Borůvka. O jistém problému minimálním (about a certain minimal problem). *Práca Moravské Přirodovědecké Společnosti*, 3:37–58, 1926. (In Czech.).

[2] B. Chazelle. A minimum spanning tree algorithm with inverse-Ackermann type complexity. *J. ACM*, 47:1028–1047, 2000.

[3] B. Chazelle. The soft heap: an approximate priority queue with optimal error rate. *J. ACM*, 47:1012–1027, 2000.

[4] D. Cheriton and R. E. Tarjan. Finding minimum spanning trees. *SIAM J. Comput.*, 5:724–742, 1976.

[5] B. Dixon, M. Rauch, and R. Tarjan. Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM J. Comput.*, 21:1184–1192, 1992.

[6] M. Fredman and D. E. Willard. Trans-dichotomous algorithms for minimum spanning trees and shortest paths. *J. Comput. Syst. Sci.*, 48:424–436, 1994.

[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] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6:109–122, 1986.

[9] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Ann. Hist. Comput.*, 7:43–57, 1985.

[10] V. Jarník. O jistém problému minimálním (about a certain minimal problem). *ráca Moravské Přirodovědecké Společnosti*, 6:57–63, 1930.

[11] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM*, 42:321–328, 1995.

[12] V. King. A simpler minimum spanning tree verification algorithm. *Algorithmica*, 18:263–270, 1997.

[13] J. Komlós. Linear verification for spanning trees. *Combinatorica*, 5:57–65, 1985.

[14] J. B. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proc. Amer. Math. Soc.*, 7:48–50, 1956.

[15] S. Pettie and V. Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49:16–34, 2002.

[16] R. C. Prim. Shortest connection networks and some generalizations. *Bell. Syst. Tech. J.*, 36:1389–1401, 1957.

[17] R. E. Tarjan. Efficiency of a good but not linear set-union algorithm. *J. ACM*, 22:215–225, 1975.

[18] R. E. Tarjan. Applications of path compressions on balanced trees. *J. ACM*, 26:690–715, 1979.

[19] A. Yao. An $O(|E| \log \log |V|)$ algorithm for finding minimum spanning trees. *Inf. Process. Lett.*, 4:21–23, 1975.