Information Dissemination in Weighted Trees

Lei Xiaoqiang
(Department of Mathematics, Kunming Teachers College, Kunming 650031, China)

Abstract: Given a weighted tree \( T = (V, E, c) \) and a fixed vertex \( r \in V \) which has some information, how to disseminate it to all vertices as soon as possible is a problem that can be solved effectively in polynomial time.

Key words: algorithm; broadcast time; tree

The broadcasting time \( b(v; T) \) of vertex \( v \) in tree \( T \) is the shortest time to propagate a message to the other vertices of tree \( T \) from vertex \( v \). The paper gives a polynomial-time optimal algorithm to compute broadcasting time \( b(v; T) \). We will use the standard notation and terminology on graphs.

2 The algorithm to compute the broadcasting time

The following is the algorithm to compute the broadcasting time. In fact, the implementation of the algorithm designs an optimal schedule.

Input: \( T = (V, E, c) \), \( r \in V \).

Output: \( t(r) \).

Step 1: \( X \leftarrow \{ \text{leaves of } T \}; \) for \( v \in V \) do \( t(v) \leftarrow 0 \) od;

\( Y \leftarrow X; \) \( T' \leftarrow T - X; \)

Step 2: while \( |V(T')| \geq 2 \) do \( X \leftarrow \{v | d(v) \leq 1, v \in V(T')\}; \) for \( v \in V \) do

\[ t(v) \leftarrow \min\{t(u) + c(u, v) : u \in X\}; \]

od; \( Y \leftarrow Y - X; \) \( T' \leftarrow T' - X; \)

The broadcasting time \( b(v; T) \) of vertex \( v \) in tree \( T \) is the shortest time to propagate a message to the other vertices of tree \( T \) from vertex \( v \). The paper gives a polynomial-time optimal algorithm to compute broadcasting time \( b(v; T) \). We will use the standard notation and terminology on graphs.

2 The algorithm to compute the broadcasting time

The following is the algorithm to compute the broadcasting time. In fact, the implementation of the algorithm designs an optimal schedule.

Algorithm

Input: \( T = (V, E, c) \), \( r \in V \).

Output: \( t(r) \).

Step 1: \( X \leftarrow \{ \text{leaves of } T \}; \) for \( v \in V \) do \( t(v) \leftarrow 0 \) od;

\( Y \leftarrow X; \) \( T' \leftarrow T - X; \)

Step 2: while \( |V(T')| \geq 2 \) do \( X \leftarrow \{v | d(v) \leq 1, v \in V(T')\}; \) for \( v \in V \) do

\[ t(v) \leftarrow \min\{t(u) + c(u, v) : u \in X\}; \]

od; \( Y \leftarrow Y - X; \) \( T' \leftarrow T' - X; \)
set the labeled adjacent vertices of \( v \) be \( u_1, \ldots, u_k \in Y \), and 
\[
t(v) \leftarrow \max_{1 \leq i \leq k} \{ t(u_i) + \sum_{j=1}^{i} c(v, u_j) \}; \text{ od}
\]
\[
Y \leftarrow Y \cup X; T' \leftarrow T' \setminus X \text{; od}
\]

**Step 3** set the labeled adjacent vertices of \( r \) be 
\( u_1, \ldots, u_k \in Y \), and 
\[
t(r) \leftarrow \max_{1 \leq i \leq k} \{ t(u_i) + \sum_{j=1}^{i} c(r, u_j) \}.
\]

As for the algorithm, we have the following theorem.

**Theorem** \( t(r) = b(r, T) \).

**Proof** Suppose that the adjacent vertices of \( r \) are 
\( u_1, \ldots, u_k \), whose subtrees are \( T_1, \ldots, T_k \), respectively, and such that 
\[
b(u_1, T_1) \geq \ldots \geq b(u_k, T_k).
\]

Then we can claim that 
\[
b(r, T) \leftarrow \max_{1 \leq i \leq k} \{ b(u_i, T_i) + \sum_{j=1}^{i} c(r, u_j) \}.
\]

In fact, set \( S_k \) be a symmetry group of the set \( \{1, \ldots, k\} \), so we have 
\[
b(r, T) \leftarrow \min_{\pi \in S_k} \{ \max_{1 \leq i \leq k} \{ b(u_i, T_i) + \sum_{j=1}^{i} c(r, u_j) \} \}.
\]

Now we only need to prove that if the permutation \( p(i) = i \) for all \( i \) in the set \( \{1, \ldots, k\} \), formula (2) can be reached. So formula (1) is true.

In fact, set \( b_i = b(u_i, T_i) \), and \( c_{ij} = c(r, u_i) \). If the permutation \( p(i) = i \) for all \( i \) in the set \( \{1, \ldots, k\} \), set the maximum in the formula (2) be 
\[
b_i + \sum_{1 \leq i \leq k} c_{ij}.
\]

From the identical permutation, via many times exchanges, we can transform it to any fixed permutation. Exchange \( (i, j) \) of position \( I \) can be classified as the following two kinds

1) \( I < j \), then on the position \( I \), we have 
\[
b_i + c_{ij} + c_{i+1} + \ldots + c_{j-1} + c_{j+1} + \ldots + c_{k} \geq n
\]
\[
b_i + c_{ij} + c_{i+1} + \ldots + c_{j-1} + c_{j+1} + \ldots + c_{k} = (3).
\]

2) \( I > j \), then on the position \( j \), we have 
\[
b_i + c_{ij} + c_{i+1} + \ldots + c_{j-1} + c_{j+1} + \ldots + c_{k} \geq n
\]
\[
b_i + c_{ij} + c_{i+1} + \ldots + c_{j-1} + c_{j+1} + \ldots + c_{k} = (3).
\]

So there is other non-identical permutation such that formula (2) is smaller than formula (3). Then formula (1) is true.

On the other hand, by formula (1) and the algorithm, we can see \( t(u_i) = b(u_i, T_i) \). This completes the proof of theorem.

Now, let us analyze the complexity of time. In algorithm, finding leaves of tree \( T \) needs time \( O(\log |V|) \), finding leaves of all trees \( T' \) needs time \( O(\log |V|) \), all of sorting needs time \( O(\log |V|) \), and all of additive computation needs time \( O(\log |V|) \), then the time complexity of algorithm is \( O(\log |V|) \). If we choose a proper data structure, we can believe that the time complexity of algorithm can decrease to \( O(\log |V|) \).

Using the algorithm to implement figure 1, we can get figure 2.

---

**Fig.1** Tree-like organization

**Fig.2** Implementation of the algorithm of figure 1

---

References:
