2003年日本オペレーションズ・リサーチ学会
秋季研究発表会

An $O(n \log^2 n)$ Algorithm for the Optimal Sink Location Problem on Dynamic Tree Networks

02602474 Osaka University *MAMADA Satoko
01013640 National Institute of Informatics UNO Takeaki
01605984 Osaka University MAKINO Kazuhisa
01502254 Kyoto University FUJISHIGE Satoru

1 Introduction

A dynamic network includes transit times on edges. We present a compound problem of a dynamic network flow and a sink location in a tree network. A location problem based on a dynamic flow is a variation of the quickest transshipment problem which is to send exactly the right amount of flow out of each source and into each sink in the minimum overall time. Hoppe and Tardos [1] presented the first polynomial-time algorithm for the quickest transshipment problem. However, their algorithm is not efficient enough. Hence, in this paper, we consider the problem in a simpler network of tree structure. This problem can be regarded as a dynamic flow version of the 1-center problem in a tree network.

We adopt a sophisticated data structure, an interval tree. We show that by using interval trees the sink location problem can be solved in $O(n \log^2 n)$ time which improves upon the previous results [2]. Here, n is the number of vertices in the network.

2 Problem Description

We consider a dynamic tree network $\mathcal{N} = (T = (V, E), c, \tau, d)$, where V is a set of vertices, E is a set of edges, $c : E \to \mathbf{R}_+$ is the upper bound for the rate of flow that enters each edge per unit time, $\tau : E \to \mathbf{R}_+$ is a transit time function, and $d : V \to \mathbf{R}_+$ is a supply function. Here, $\mathbf{R}_+$ denotes the set of all nonnegative reals.

The problem to be considered here is to find a sink $t \in V$ such that we can send given initial supplies $d(v)$ $(v \in V \setminus \{t\})$ to sink t as quick as possible. Suppose that we are given a sink t in T. Then, T is regarded as an in-tree $\vec{T}(t) = (V, \vec{E}(t))$ with root t, i.e., each edge of T is oriented toward the root t. For any arc $e \in \vec{E}(t)$, any $\theta \in \mathbf{R}_+$, we denote by $f_e(\theta)$ the flow rate entering arc e at time θ which arrives at the head of e at time $\theta + \tau(e)$. We call $f_e(\theta)$ $(e \in \vec{E}(v^*),\ \theta \in \mathbf{R}_+)$ a *continuous dynamic flow* in $\vec{T}(v^*)$ (with a sink v^*) if it satisfies the following three conditions; (1) capacity constraints, (2) flow conservation, and (3) demand constraints.

For a continuous dynamic flow f, let θ_f denote the completion time for f and let $C(v^*)$ denote the minimum θ_f among all continuous dynamic flows f in $\vec{T}(v^*)$. We study the problem of computing a sink $v^* \in V$ with minimum $C(v^*)$.

3 Algorithm

In the algorithm, we keep two tables, *Arriving Table* A_v and *Sending Table* S_v for each vertex $v \in V$. Arriving Table A_v represents the sum of the flow rates arriving at the vertex v as a function of time θ. Sending Table S_v represents the flow rate leaving the vertex v as a function of time θ. We describe Algorithm Single-Phase which is simpler than the algorithm proposed in [2].

Intuitively, our algorithm first constructs Arriving Tables A_v for all leaves v. Then we find a leaf v^* which is not an optimal sink (more precisely, a leaf v^* such that T has an optimal sink other than v^*), and remove it from T. If some vertex v becomes a leaf of the modified tree T, then the algorithm computes Arriving Table A_v for this vertex v by using Arriving tables for the vertices that are adjacent to v and have already been removed. The algorithm repeatedly applies this procedure to T until T becomes a single vertex t, and outputs such a vertex t as an optimal sink.

For adjacent vertices v and $p(v)$ in T, deleting edge $\{v, p(v)\}$ from T yields two connected components. Denote by $T(v, p(v))$ the component containing v and by $T(p(v), v)$ the one containing $p(v)$. $Time(v, p(v))$ represents the completion time in which all the initial supplies $d(v)$ ($v \in T(v, p(v))$) can be sent to $p(v)$ as quick as possible.

Algorithm SINGLE-PHASE

Input: A tree network $\mathcal{N} = (T = (V, E); c, \tau, d)$.

Output: An optimal sink t that has the minimum completion time $C(t)$ among all vertices of T.

Step 0: Let $W := V$, and let L be the set of all leaves of T. For each $v \in L$, construct Arriving Table A_v.

Step 1: For each $v \in L$, construct Sending Table S_v from v to $p(v)$ based on ceiling A_v by $c(v, p(v))$, where $p(v)$ is a vertex adjacent to v in T. Compute the time $Time(v, p(v))$.

Step 2: Compute a vertex $v^* \in L$ such that $Time(v^*, p(v^*)) = \min_{v \in L} Time(v, p(v))$. Let $W := W \setminus \{v^*\}$, $L := L \setminus \{v^*\}$.

If there exists a leaf v of $T[W]$ such that v is not contained in L,

then :

Let $L := L \cup \{v\}$. Construct Arriving Table A_v based on adding Sending Table $S_{v'}$ shifted by $\tau(v', v)$ for the vertices v' that are adjacent to v in T and have already been removed from W.

Compute Sending Table S_v from v to $p(v)$ based on A_v, where $p(v)$ is a vertex adjacent to v in $T[W]$.

Compute the time $Time(v, p(v))$.

Step 3: If $|W| = 1$, then output $t \in W$ as an optimal sink. Otherwise, return to Step 2.

□

Note that in Step 2, at most one leaf v of $T[W]$ is not contained in L, and L is always the set of all leaves of $T[W]$ after Step 2.

4 Data structures for A_v and S_v

We consider data structure for Arriving Table A_v and Sending Table S_v. Algorithm SINGLE-PHASE requires $O(n^2)$ time if explicit representations are used for A_v and S_v. Therefore, we need sophisticated data structures for them so that we can efficiently handle three basic operations, *Add-Table* (i.e., adding tables), *Shift-Table* (i.e., shifting a table), and *Ceil-Table* (i.e., ceil a table by some capacity c). We adopt interval trees to represent tables, which are standard data structures for a set of intervals, since our tables can be regarded as sets of intervals. It is known that interval trees can handle operations *Add-Table* and *Shift-Table* efficiently. However, interval trees do not seem to handle operation *Ceil-Table* efficiently if we implement interval trees straightforwardly. We develop a method to represent those tables implicitly. The method can handle all the three operations efficiently. Although we skip the details, by applying it to algorithm SINGLE-PHASE, we have an $O(n \log^2 n)$ time algorithm.

Theorem 4.1: The sink location problem on dynamic tree networks can be solved in $O(n \log^2 n)$.

5 Concluding Remarks

We have described our result on an algorithm for quickest flows in a tree network. Finally, we note that the sink location problem for dynamic flows can further be extended in many directions. Some of them are (1) to find a sink to which we can send a flow of maximum value from sources within given fixed time, (2) to consider the sink location problem on general (non-tree) dynamic networks, and (3) to consider a multiple-sink location problem. These are left for future research.

Acknowledgment

This research was supported by the Japan Society for the Promotion of Science under Grant-in-Aid for Creative Scientific Research (Project No. 13GS0018).

References

[1] B. Hoppe and É. Tardos: The quickest transshipment problem, *Mathematics of Operations Research*, **25** (2000) 36–62.

[2] S. Mamada, K. Makino and S. Fujishige: Optimal sink location problem for dynamic flows in a tree network, *IEICE Trans. Fundamentals*, **E85-A** (2002), 1020–1025.