ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 5, Issue 4, April 2016

Design of algorithms to Maximum flow Problems in tree flow Networks

A. Anto Kinsley, B. Uma Maheswari

Abstract **— In this paper we introduce a network, called a tree flow network, which is a deduction of flow network with some constraints. Maximum flow problems involve finding a feasible flow through a flow network. We can model a drinking water supply system as a tree flow network. We also discuss tree flow network and design algorithms for the maximum flow problem of the network. The speed of their algorithm, it turns out, depends on the edge capacities in the network as well as on the numbers *V* of vertices and *E* of edges of the network.**

Index Terms **—Algorithm, capacity, flow, maximum flow, Networks.**

I. INTRODUCTION

For graph theoretic terminology, we refer to Bandy and Murthy [1]. The network flow problem [2] is an example of a beautiful theoretical subject that has many important applications. In optimization theory, maximum flow problems involve finding a feasible flow through a single-source, single-sink flow network that is maximum. The first algorithm for the network flow problem was given by *Ford* and *Fulkerson* [3]. They used that algorithm not only to solve instances of the problem, but also to prove theorems about network flow. In particular, they used their algorithm to prove the '*max-flow-min-cut*' theorem [3].

Definition 1.1

A *network* is an edge-capacitated directed graph with two distinguished vertices called the source and the sink. The words source for *s* and sink for *t* suggest at *s* has zero in degree and *t* has zero out degree. This need not be the case. However, we shall see later that the arcs entering or those leaving *t* or actually redundant and do not affect network flow problems. We allow these arcs, though, since the choice of *s* and *t* may be changed for the same network.

We can think of the edges of *G* as conduits for a fluid, the capacity of each edge being the carrying, capacity of the edge for that fluid. The fluid flows in the network from the source to the sink, in such a way that the amount of fluid in each edge does not exceed the capacity of that edge.

Definition 1.2

A *flow* in a network X is a function f that assigns to each edge *e* of the network a real number *f*(*e*), in such a way that

(1) For each edge *e* we have o ≤ *f*(*e*) ≤ cap(*e*) and
(2) For each vertex v other than the source and the sink, it is true that $\sum_{init(e)=v} f(e) = \sum_{Term(e)=v} f(e)$ _____ (1)

The condition (1) is a flow conservation condition. It states that the outflow from *v* is equal to the inflow to *v* for all vertices *v* other than *s* and *t*.

2. The tree flow networks

Definition 2.1

A *tree flow network* is a directed flow network provided that there is a unique path between any pair of vertices and the end nodes are labeled as the sink *t*. We can think of the edges of *G* as conduits for a fluid, the capacity of each edge

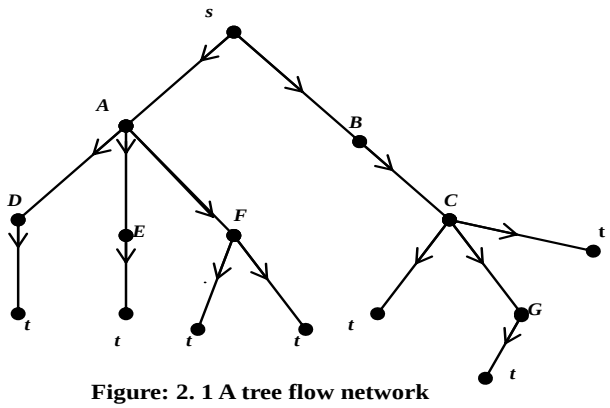


Figure: 2. 1 A tree flow network

being the carrying, capacity of the edge for that fluid. The fluid flows in the network from the source to the sink, in such a way that the amount of fluid in each edge does not exceed the capacity of that edge.

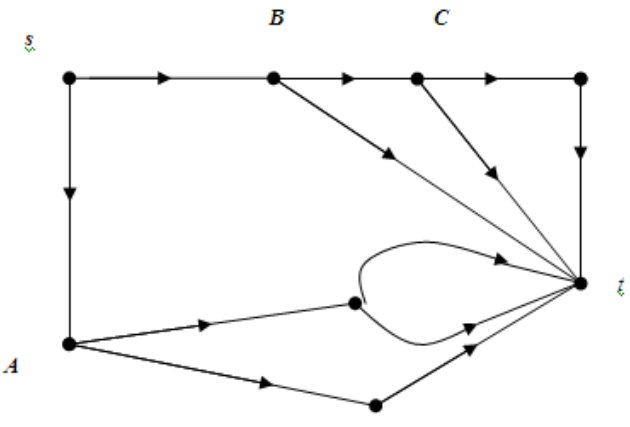


Figure: 2. 2 A Tree Flow Network

The underlying digraph *D* of a tree network is a tree, that is, for distinct vertices *u* and *v* of *D*, not both (*u*, *v*) and (*v*, *u*) are

arcs of *D*. The network shown in the figure 2.1 can be viewed like the above figure.

Modified Ford Fulkerson algorithms

If an edge $e = (u, w)$, when $v \neq w$, directed from vertex v to vertex w, then v is the initial vertex of e and w is the terminal vertex of e. We may then write v = Init (e) and w = Term (e). No edge (w, v) is permitted. A tree flow network is shown in figure 2.1 and it can be drawn as a network shown in figure 2.2.

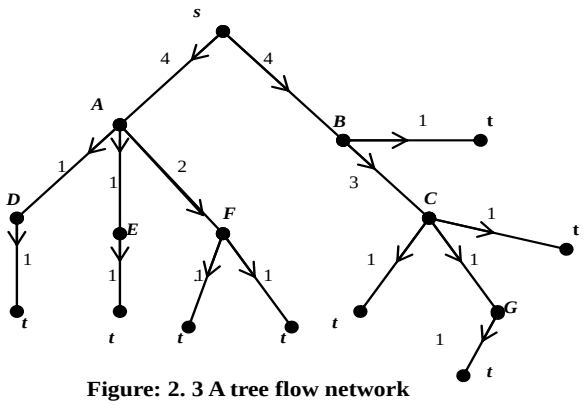


Figure: 2. 3 A tree flow network

No edge (w, v) is permitted. A tree flow network is shown in figure 2.1 and it can be drawn as a network shown in figure 2.2.

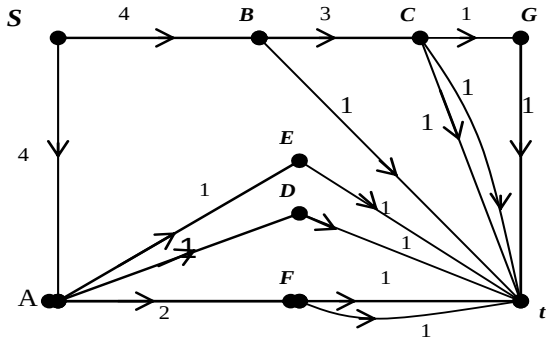


Figure: 2. 4: A tree flow network

Definition 2.2

A flow in a network X is a function f that assigns to each edge e of the network a real number $f(e)$ in such a way that for each edge e we have $0 \leq f(e) \leq \text{cap}(e)$ and for each vertex v other than s and t it is true that $\sum_{y \in N^+(x)} f(x,y) = \sum_{y \in N^-(x)} f(y,x)$ for every vertex other than s and t. Let Q be the net out flow from s then Q is also the net inflow to t. The quantity Q is called the value of the flow.

ALGORITHMS DEVELOPMENTS FOR THE FLOW PROBLEM

The basic idea of the algorithms for the network flow problem is this: start with some flow function (initially this might consist of zero flow on every edge). Then look for a flow augmenting path in the network. A flow-augmenting path is a path from the source to the sink along which we can push some additional flow. An edge can get selected to a flow-augmenting path for the reason that the value of the flow function on the edges is below the capacity of that edge. Then we can increase the flow along the edge and on all edges. It is, of course, necessary to maintain the conservation of flow. To do this we will augment the flow on every edge of an augmenting path by the same amount. The algorithm first finds a flow augmenting path. Then it augments the flow along that path as much as it can. Then it finds another flow augmenting path etc, etc. The algorithm terminates when no flow augmenting paths exists. The flow with then be at the maximum possible value.

Definition: 2. 3

Let f be a flow function in a network X. We say that an edge e of X is usable from v to w if either e is directed from v to w and the flow in e is less than the capacity of the edge. Given a network and a flow in that network, we find a flow-augmenting path from the sources to the sink. This is done by a process of labeling and scanning the vertices of the network, beginning with the source and proceeding out to the sink. Initially all vertices are in the conditions 'unlabeled' and 'un scanned'. As the algorithm proceeds, various vertices will become labeled and if a network is labeled, it may become scanned. To scan a vertex v means that we stand at v and look around at all neighbors w of v that haven't yet been labeled. If e is some edge that joins v with a neighbor w, and if the edge e is usable from v to w as defined above, then we will label w, because any flow augmenting path that has already reached from the source to v can be extended another step, to w.

The label that every vertex v gets is a pair (u, z) where 'u' part of the label of v is the name of the vertex that was being scanned when v was labeled. Finally, the 'z' component of the label represents the largest amount of flow that can be pushed from the source to the present vertex v along any augmenting path that has so for been found. At each step the algorithm will replace the current value of z by the amount of new flow that could be pushed through to z along the edge that is now being examined, if that amount is smaller than z. To begin with, the algorithm labels the source with (-∞,∞). The source now has the label-status labeled and the scan-status un-scanned. Next we will scan the source. Here is the procedure for scanning any vertex u.

Algorithm: 2.4

Procedure scan (u: vertex; X: network; f: flow);
For every 'unlabeled' vertex v that is connected to u by an edge
do
If the flow in (u, v) is less than cap (u, v)
Then label v with (u, min {z (u), cap (u, v) – flow (u, v)});
Change the label-status of v to 'labeled';
Change the scan-status of u to 'scanned';
End.

We can use the above procedure to describe the complete scanning and labeling of the vertices of the network, as follows.

Algorithm: 2.5
Procedure label and scan(*X*: network; *f*: flow; why halt: reason);
Give every vertex the scan-status 'un-scanned' and the label-status 'unlabeled';
U: = source;
Label source with (-∞,∞);
Label –status of source = 'labeled';
While {there is a 'labeled' and 'un-scanned' vertex *v* and sink is 'unlabeled'}
Do scan (*v*, *X*, *f*);
If sink is unlabeled
Then 'why halt' = 'flow is maximum'
Else 'why halt' = 'it is time to augment'
End.

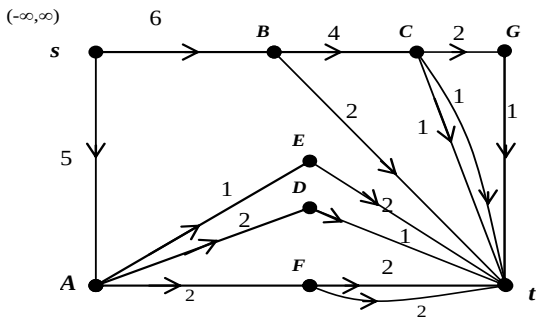
The flow-augmenting algorithm is the following.

Algorithm: 2.6
Procedure augmenting flow(*x*: network; *f*: flow; amount: real);
{Assumes that label and scan has just been done)
V: = sink;
Amount: = the '*z*' part of the label of the sink;
Repeat
(Previous, *z*):= label (*v*)
Increase f (previous, *v*) by amount;
V: = previous
Until *v* = source;
End.

The value of the flow in the network has now been increased by *z* units. The whole process of labeling and scanning is now repeated, to search for another flow augmenting path. The algorithm halts only when we are unable to label the sink.

Algorithm: 2.7
Procedure max flow (*X*: network; *f*: flow; max flow: real);
{Finds maximum flow in a given network *X*}
Set *f*: = 0 on every edge of *X*;
Max flow value: = 0;
Repeat
Label and scan (*X*, *f*, why halt);
If why halt = 'it is time to augment' then
Augment flow (*X*, *f*, amount);
Max flow value: = max flow value + amount
Until why halt = 'flow is maximum'
End.

Let *V* be the number of vertices, V_0 be the number of end vertices and *E* be the number of edges of the tree flow network. Each iteration of the while loop takes *O* (*E)* time. Then the above algorithms work with *O* (*E* V_0) time. That is *O* (*E V*) time complexity. The above algorithms can be applied to find the maximum flow pushed through any tree flow network. Consider the above network for finding the maximum flow applied.



Iteration- **Figure: 2. 5 Augment path along (*S, B, t*)**

Augment flow along (*S*, *B*, *t*)

v = sink = *t*

Amt = the '*z*' part of the label of the sink

Using flow augmenting algorithm, label (*t*) = (*B*, 2)

Therefore, the flow of (*B*, *t*) is increased by 2 until *v* = source.

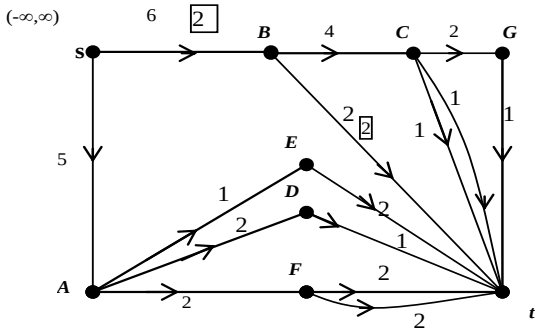


Figure: 2. 6 Augment path along (*S, B, t*)

Iteration-2

Augment flow along (*S*, *B*, *c*, *t*) which occurs 2 times.

Label (*t*) = (*C*, 1)

Therefore, applying the algorithm, the maximum flow along (*S*, *B*, *C*, *t*) is increased by 1 along the 2 paths.

Iteration-3

Augment flow along (*S*, *B*, *C*, *G*, *t*)

Label (*t*) = (*G*, 1). Therefore, applying the algorithm the maximum flow along (*S*, *B*, *C*, *G*, *t*) is increased by the maximum flow 1.

Iteration-4

Augment flow along (*S*, *A*, *E*, *t*)

Label (*t*) = (*E*, 1). Therefore, applying the algorithm the maximum flow along (*S*, *A*, *E*, *t*) is increased by the 1.

Iteration -5

Augment flow along (*S*, *A*, *D*, t)

Label (*t*) = (*D*, 1)

Therefore, applying the algorithm the maximum flow along (*S*, *A*, *D*, *t*) is increased by the 1

Iteration-6

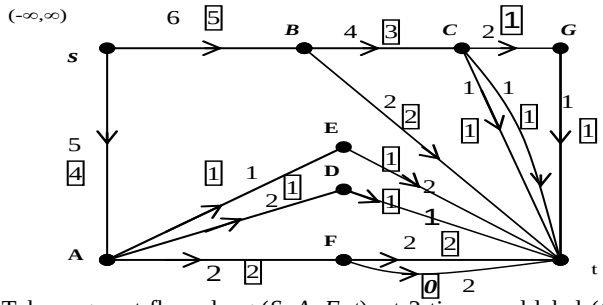


Figure: 2. 7 Network with max flow

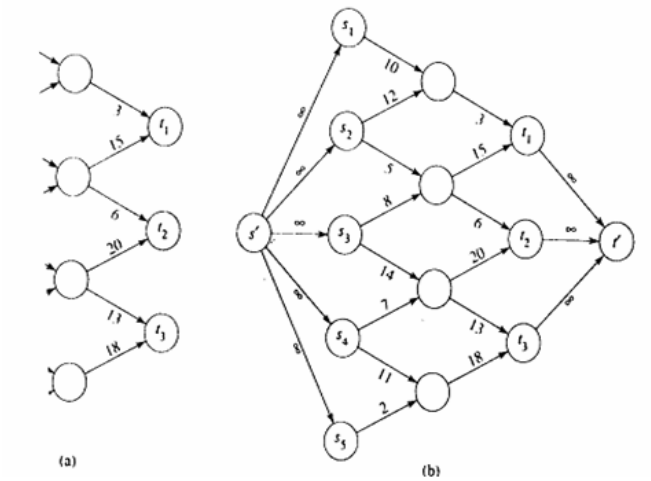
Take augment flow along (*S, A, F, t*), at 2 times and label (*t*) (*F*, 2).Applying , *t*) is increased by 2 in one way.

In another way the flow along (*F, t*) is 0 since the flow value cannot exceed the capacity value. Now the algorithm terminates since why halt = 'flow is maximum'. By max-flow min-cut theorem the maximum flow is 9 units and the resultant network is given in figure 2.7.

Maximum flow can be pushed through the network is
5 + 4 = **9**.

Network with multiple sources and sinks

A maximum – flow problem can have several sources and sinks. For example a company has a set of m factories (s_1, s_2,…, s_m) and a set of is ware house $\{t_1, t_2, \ldots..,t_n\}$ as in the following figure



We can reduce the problem of determining max flow in a network with multiple sources and sinks to ordinary max flow problems.

We add a super source *s* and add a directed edge(s, s_i) with capacity $c(s, s_i) = \infty$ for each $i = 1, 2\ldots, m$. We also create a new super sink t and add a directed edge (t_j, t) with capacity c (t_j; t) =∞, for each i = 1, 2… n. The single source *s*, simply provides as much flow as desired for the multiple sources s_i, and the single sink t likewise consumer as much flow as desired for the multiple sinks t_i. Using the above algorithms the maximum flow can be found for the network with multiple sources and sinks.

REFERENCES

[1] Bandy. J. A, & Murthy, U. S. R. (1976). *Graph theory with applications*, Elsevier Science Publishing co. Inc., U. S. A.
[2] Abuja, R. K.., Magnate T. L., & Orin, J. B. (1993). *Network Flow*, Englewood Cliffs, N. J. Prentice-Hall. Goldberg, A. V., &. Trajan, R. E. (1988) *A New Approach to the Maximum Flow Problem*, J. ACM 35,
[3] Ford, L.R., & Fulkerson, D.R. (1974). *Flows in networks*, Princeton, NJ, 921- 940.
[4] Goldberg, A. V., &. Trajan, R. E. (1988) *A New Approach to the Maximum Flow Problem*, J
[5] Wolf, H. S. (1986). *Algorithms and Complexity*, U.S.A. Prentice – Hall International, Inc,

Author's Profile and Image

Name: A. Anto Kinsley,
Qualification: M.Sc., M. Phil, M. Tech., Ph. D.
Area of Specialization: Graph Algorithms, Flow Network, Distance in Graphs, Central Structures In Graphs, Domination and Distance in graphs, Star graphs, Convexity In Graphs, DNA Computation.
Address for Communication: 20.B, St. Paul's Street, Palayamkottai. Tirunelveli.
Mobile Number: 9443446496.
Number of publications in International journals: 19

Name: B. Uma Maheswari,
Qualification: M.Sc., M. Phil, PGDCA
Area of Specialization: Graph Theory and Algorithms,
Address for Communication: 119, Big Street, Tirunelveli Town.
Mobile Number: 9442624315.
Number of publications in International journals: 2