ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 4, Issue 1 (Jan. - Mar. 2016)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2016)

Design of Algorithms to Generalized Maximum Flow Network Problem

[I]Anto Kinsley, [II]Uma Maheswari
[I]Dept. of Mathematics, St. Xavier's College (Autonomous), Palayamkottai. Tamilnadu, India
[II]Research Scholar, Dept. of Mathematics, St. Xavier's college (Autonomous), Palayamkottai. Tamilnadu, India

Abstract
This work presents an algorithm for the generalized maximum flow problem. First, we describe the traditional maximum flow problem, presented by Ford and Fulkerson, there is a capacitated network and the goal is to send as much flow as possible between two distinguished nodes, without exceeding the arc capacity limits. In traditional network, there is an implicit assumption that flow is conserved on every arc. In generalized networks, each arc has a positive multiplier γ(u, v) called a gain factor, associated with it, representing the fraction of flow that remains when it is sent along that arc. The generalized maximum flow problem is identical to the traditional maximum flow problem, except that it can also model network which "leak" flow.

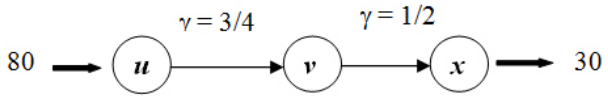
Keywords
Networks, Flow, Capacity, Gain Function, Maximum Flow, Generalized Network.

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. 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].

The generalized maximum flow problem is a natural generalization of the traditional maximum flow problem. In traditional networks, there is an implicit assumption that flow is conserved on every arc. This assumption may be violated if water leak as it is pumped through a pipeline. Many applications are described in [4].

In a generalized network, a fixed percentage of the flow is lost when it is sent along an arc. Specially, each arc (u, v) has an associated gain factor $\gamma(u, v)$. When $f(u, v)$ units of flow enter into the arc (u, v) through node u then $\gamma(u, v) f(u, v)$ arrive at v. As the example in Figure illustrates, if 80 units of flow are sent into an arc (u, v) with gain factor 3/4, then 60 units reach node v; if these 60 units are then sent through an arc (v, x) with gain factor 1/2, then 30 units arrive at x.



A. Definition

A *network* is an edge capacitated directed graph with two distinguished vertices, source s and sink t. We can think of the edges of G as conduits for a fluid, and the capacity of each edge being 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.

B. Definition

An instance of the maximum flow problem is a network $G=(V, E, s, t, c)$ where $s \in V$ is distinguished node called the *source*, $t \in V$ is a distinguished nodes called the *sink* and c is a capacity function. A *flow* is a **pseudo-flow** that satisfies and the *flow conservation constraints*:

$$\forall v \in V - \{s,t\}: \sum_{w \in V:(v,w) \in E} f(v,w) = 0$$

The *value* of a flow f is the net flow into the sink:

$$|f| = \sum_{v \in V:(v,t) \in E} f(v,t)$$

The objective is to find a flow of maximum value.

II. Generalised Maximum Flow Problem

A. Definition

The generalized maximum flow problem is a *generalized network* $G = (V, E, t, c,\gamma,e)$ where V is an n-set of nodes, E is an m-set of directed arc, $t \in V$ is a distinguished node called the sink, $c : E \to R \geq 0$ is a capacity function, $\gamma : E \to R \geq 0$ is a *gain function*, and $e : V \to R \geq 0$ is an *initial excess function*. A *residual arc* is an arc with positive capacity. A *flow generating cycle* is a cycle whose gain is more than one.

B. Definition

A *generalized pseudo flow* is a function $f : E \to R$ that satisfies the *capacity constraints*

$f(v,w) \leq u(v,w)$ for all $(v,w) \in E$ and the *anti symmetry constraints* $f(v,w) = -\gamma(w,v) f(w,v)$ for all $(v,w) \in E$. The *residual excess* of f at node v is $e_f(v) = e(v) - \sum_{(v,w) \in E} f(v,w)$ i.e., the initial excess minus the net flow out of v. If $e_f(v)$ is positive (negative) we say that f has residual excess (deficit) at node v. A *generalized flow* is a generalized pseudo flow that has no residual deficits, but it is allowed to have residual excesses. A *proper generalized flow* is a flow which does not generate any additional residual excess, except possibly at the sink.

We note that a flow can be converted into a proper flow, by removing flow on useless paths and cycles. Let $OPT(G)$ denote the maximum possible value of any flow in network G. A flow f

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

is *optimal* in network G if $|f| = OPT(G)$ and ξ- *optimal* if $|g| \geq (1 - \xi)\ OPT(G)$. The (*approximate*) *generalized maximum flow problem* is to find a (ξ-) optimal flow.

Optimality conditions

An *augmenting path* is a residual path from a node with residual excess to the sink. A *generalized augmenting path* (GAP) is a residual flow-generating cycle, together with a (possibly trivial) residual path from a node on this cycle to the sink. By sending flow along augmenting paths or GAPs we increase the net flow into the sink.

C. Theorem

A flow f is optimal in network G if and only if there are no augmenting paths or GAPs in G_f

Proof: Refer [5].

III. Modified Ford Fulkerson Algorithm

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 elected 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.
To begin with, the algorithm labels the source with (-∞,∞). The source now has the label-status labelled and the scan-status unscanned. Next we will scan the source. Here is the procedure for scanning any vertex *u*.

A. Algorithm

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 'labelled';
Change the scan-status of *u* to 'scanned';
End.
We can use the above procedure to describe the complete scanning and labelling of the vertices of the network, as follows

B. Algorithm

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 = 'labelled';
While {there is a 'labelled' 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.

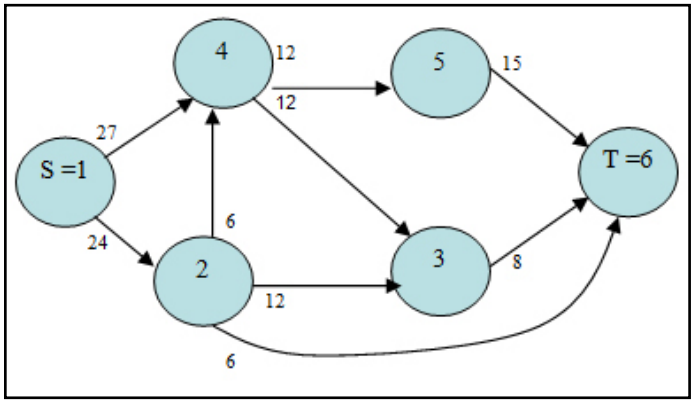
C. Algorithm

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 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 labelling and scanning is now repeated, to search for another flow augmenting path. The algorithm halts only when we are unable to label the sink.

D. Algorithm

Procedure max – in - flow (*X*: network; *f*: flow;
max-in-flow: real);
{Finds maximum in flow in a given network *X*}
Set *f*: = 0 on every edge of *X*;
Max – in - 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 – in - flow value: = max-in-flow value
+ Amount
Until why halt = 'flow is maximum'
End.
The above algorithm can be applied to find the maximum in flow. Consider the following network for finding the maximum in flow applied.



I. iteration

Augment flow along (1, 4, 5, 6)
v = sink = *t*
Amt = the '*z*' part of the label of the sink
Using flow augmenting algorithm, label (*t*) = (4, **12**)
Therefore, the flow of (*4*, 5) is increased by 12 until
v = source.

II iteration

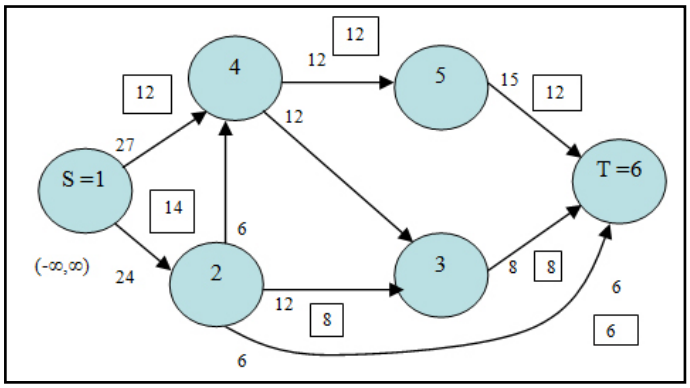
Augment flow along (1, 2, 3, 6).

Label $(t) = (3, \mathbf{8})$. Therefore, applying the algorithm, the maximum flow along (1, 2, 3, 6) is increased by 8.

III iteration

Augment flow along (1, 2, 6)

Label $(t) = (2, \mathbf{6})$. Therefore, applying the algorithm the maximum flow along (1, 2, 6) is increased by 6.



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 in flow is 26 units and the resultant network is given in the above figure.

Maximum in flow = 12 +14 = 26.

We shall present an algorithm sends flow along all gain augmenting paths simultaneously, using a maximum flow computation the algorithm is given below.

E. Algorithm: 3.5

Procedure max flow (X: network; f: flow;
Gain function$_{\gamma : E \to R}$: max flow: real)
{Finds maximum out flow in a given network}
Max -out-flow value: = 0
Set $\gamma = 1 - \xi, where 0 < \xi < 1$
While there exists an augmenting path do
Max – out - flow value: = max – out - flow value + γ

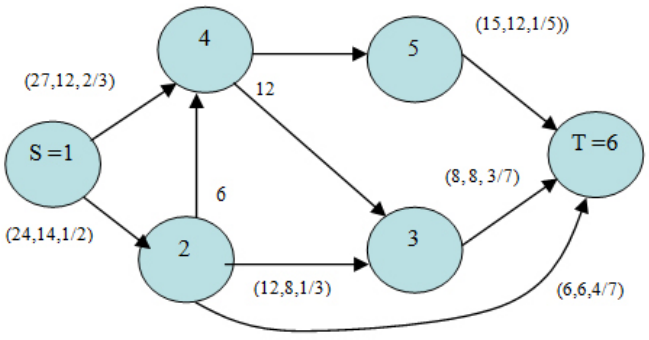
Where $\gamma = \sum\gamma(x) + \sum\gamma(y)$

End {while}.

To find the maximum out flow

Consider the network with gain function.

(12, 12,1/12)



In this network we have GAP:

F_1: 1 – 4 – 5 – 6; F_2: 1 – 2 – 3 - 6 and 1 – 2 – 6.

Let x and y be the gain function of the corresponding paths F_1 and F_2

In F_1, The gain function of the path 1 – 4 – 5 - 6 can be expressed as

$$x \to \frac{x}{3} = 0.333x$$

$$\frac{x}{3} \to \frac{x}{3} * \frac{1}{12} * \frac{4}{5} = 0.333x * 0.0666$$

In F_2, The gain function of the 1 – 2 – 3 – 6 can be expressed as

$$y \to \frac{y}{2} = 0.5y$$

$$\frac{y}{2} \to \frac{y}{2} * \frac{2}{3} * \frac{4}{7} = 0.5y * 0.381$$

In the path 1 – 2 – 6, the expression can be written as

$$y \to \frac{y}{2} = 0.5y$$

$$\frac{y}{2} \to \frac{y}{2} * \frac{3}{7} = 0.5y * 0.429$$

Maximum out flow = sum of the gain function of x + sum of the gain function of y.

Iteration: 1

When x = 1, Augment flow along (1, 4, 5, 6)

The gain function $\frac{x}{3} = 0.333x$

0.333x*0.0666 = 0.0222

When y = 1, Augment flow along (1, 2, 3, 6)

The gain function $\frac{y}{2} = 0.5y$

0.5y * 0.381= 0.19

Augment flow along (1, 2, 6)

0.5y * 0.429 = 0.214

Maximum out flow= 0.0222 + 0.19 + 0.214 = 0.426

Iteration: 2

When x = 2, Augment flow along (1, 4, 5, 6)

The gain function $\frac{x}{3} = 0.333 * 2 = 0.667$

0.667 * 0.333 = 0.222

When y = 2, Augment flow along (1, 2, 3, 6)

The gain function $\frac{y}{2} = 0.5y = 0.5 * 2 = 1$

1* 0.381= 0.381

Augment flow along (1, 2, 6)

1* 0.429 = 0.429

Maximum out flow= 0.222 + 0.381 + 0.429 = 1.032

By iteration3, Maximum out flow = 1.215

Proceeding like this, we will increase the value of x and y, up to $x = 27$ and $y = 24$.

Result: Maximum in flow = 26

Maximum out flow = 10.32

Let V be the number of nodes, V_0 be the number of end nodes and E be the number of edges of the tree flow network.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Each iteration of the while loop takes $O(E)$ time. Then the above algorithms work with $O(E V_o)$ time. That is $O(EV)$ time complexity.

IV. Applications

In traditional networks, there is an implicit assumption that flow is conserved on every arc. Many practical applications violate this conservation assumption. The gain factors can represent physical transformations of one commodity into a lesser or greater amount of the same commodity. Some examples include: spoilage, theft, evaporation, taxes, seepage, deterioration, interest, or breeding. The gain factors can also model the transformation of one commodity into a different commodity. Some examples include: converting raw materials into finished goods, currency conversion, and machine scheduling. We explain the latter two examples next.

Currency Conversion

We use the currency conversion problem as an example of the types of problems that can be modelled using generalized flows. Later, we will use this problem to gain intuition. In the currency conversion problem, the goal is to take advantage of discrepancies in currency conversion rates. Given certain amount of one currency, say 1000 U.S dollars, the goal is to convert it into the maximum amount of another currency, say French Francs, through a sequence of currency conversions. We assume that limited amounts of currency can be treated without affecting the exchange rates.

Scheduling unrelated parallel machines

As a second example, we consider the problem of scheduling N jobs to run on M unrelated machines. The goal is to schedule all of the jobs by a pre specified time T. Each job must be assigned to exactly one machine. Each machine can process any of the jobs, but at most one job at a time. Machine i requires a pre specified amount of time P_{ij} to process job j.

References

[1] *J. A. Bandy and U. S. R. Murthy, Graph theory with applications, Elsevier Science publishing co., Inc. U. S. A. 1976.*
[2] *R. K. Abuja, T. L. Magnanti, J. B. Orin, Network Flow, Prentice Hall, Englewood Cliffs, N. J. 1993.*
[3] *L.R. Ford and D.R. Fulkerson, Flows in networks, Princeton University. Press, Princeton, N..J. 1974.*
[4] *F. Glover, J.Hultz, D. Kingman, and J. Stutz. Generalized Networks: A fundamental computer based planning Tool. Management Science, 24:1209-1220, 1978.*
[5] *H. S. Wolf, Algorithms and Complexity, Prentice Hall International, Inc., U. S. A. 1986.*

Author's Profile

Anto Kinsley, 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. E-mail: antokinsley@yahoo.com, Mobile : 9443446496.

Uma Maheswari, M.Sc., M. Phil, PGDCA. Area of Specialization: Graph Theory And Algorithms, Address for Communication: 119, Big Street, Tirunelveli Town. Email : umaraju211199@gmail.com, Mobile 9442624315.