[Kinsley*, 5(2): February, 2016]
ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A PRE-FLOW PUSH ALGORITHM TO GENERALIZED MAXIMUM FLOW PROBLEM

A. Anto Kinsley *, B. Uma Maheswari
Department of Mathematics, St. Xavier's college (Autonomous), Palayamkottai.
Research scholar, Department of Mathematics, St. Xavier's college (Autonomous), Palayamkottai.

ABSTRACT

This work presents an algorithm for the generalized maximum flow problem. First, we describe the traditional maximum flow problem. Pre-flow Push algorithms work in a more localized manner than the Ford-Fulkerson method. These algorithms maintain at all stages a feasible pre-flow that has a saturated cut. The pre-flow is changed step by step until it does satisfy flow conservation. The resulting flow then has a saturated cut so is a maximum flow. In generalized networks, each arc has a positive multiplier $\gamma(u, v)$ 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 with "leak" flow.

KEYWORDS: Network, distance labeling, excess flow, capacity, maximum flow, gain function.

INTRODUCTION

For graph theoretic terminology, we refer to Bundy and Murthy [1]. The network flow problem [2] is an example of a beautiful theoretical subject that has many important applications. In this paper, the ideas of Goldberg- Trajan [4] pre flow and distance labeling on a network will be fashioned into an algorithm for the maximum flow problem and it is called as pre-flow push algorithm. It is based on the local routine that operates on the excess flow at a single vertex.

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 [3].

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) at node u then $\gamma(u, v) f(u, v)$ arrive at v. As the example in Figure 1 illustrates, if 80 units of flow are sent into an arc (u, v) with gain factor 3/4, then 60 units reach node w; if these 60 units are then sent through an arc (v, x) with gain factor 1/2, then 30 units arrive at x.

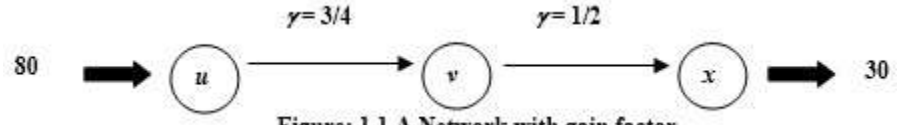


Figure: 1.1 A Network with gain factor

Definition 1.1
A *network* is an edge-capacitated directed graph with two distinguished vertices, source s and sinks 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. 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 $o \leq f(e) \leq \text{cap}(e)$ and

(2) For each vertex *v* other than the source and the sink, it is true that

$$\sum_{Init(e)=v} f(e) = \sum_{Term(e)=v} f(e) \text{ ——————— } (1)$$

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*.

Definition 1.2

A network with edge capacities $c_{a\,b}$ and $f = \{f_{a\,b}\}$ is an assignment of flow to the edges of *N*. Then *f is* a *pre-flow* if at each internal node $v \neq s, t$, we have $\sum_{h(e)=v} f_e \geq \sum_{t(e)=v} f_e$. Thus we assume that, at each internal node, the flow in is at least as large as the flow out. The *excess flow* at *v* is defined by

$$ex(v) = \sum_{h(e)=v} f_e - \sum_{t(e)=v} f_e = \sum_{w} f_{wv} - \sum_{u} f_{vu}$$

Of course, if *ex* (*v*) = 0 for all internal nodes *v* then *f is* a flow. We call *f* a feasible pre-flow if, in addition, $0 \leq f_e \leq C_e$. Our general strategy is to start with an initial pre-flow then successively modify the flows on various edges so that at all times the pre-flow is feasible and moves steadily toward satisfying flow conservation.

Definition 1.3

A *distance labeling* λ assigns to each node *v* an integer $\lambda(v)$. The conditions that makes $\lambda(v)$ a distance labeling are that: $\lambda(s) = n = |V|$, $\lambda(t) = 0$ and if edge *a b* is available for a push then $\lambda(a) \leq \lambda(b) + 1$. So we insist that, whenever $f_{a\,b} < c_{a\,b}$ or $f_{b\,a} > 0$ we have $\lambda(a) \leq \lambda(b) + 1$.

Generalized Maximum Flow problem

Definition 2.1

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.

Definition 2.2

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* is *optima*l 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 into the sink. A *generalized augmenting path* (GAP) is a residual flow-generating cycle, together with a 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.

Theorem 2.3

Suppose that network N has a feasible pre-flow P and a distance labelingλ. Then there is a cut (S, T) with every edge of δ(S) saturated and every edge of δ(T) empty.

Proof: Refer [4]
This theorem says that any feasible pre-flow with a distance labeling has a saturated cut. For the special case of a flow, the Max-Flow Min-Cut Theorem then gives the following corollary which establishes the stopping condition for the pre-flow push algorithms.

Theorem 2.4: If a feasible flow *F* has a distance labeling then *F is a maximum flow.* □

PRE -FLOW PUSH ALGORITHM

We have a network *N* with a pre flow *f and* a distance labelingλ. At each internal node *v* there is some excess flow *ex* (*v*). We denote the capacity of the arc *a b* by c_{ab} while the pre flow on this edge is f_{ab}. We call an edge *a b* an *available edge* if either *a b* or *b a* is an arc of *N* either $f_{ab} < c_{ab}$ or $f_{ba} > 0$ or both. Thus an available edge is a candidate for a push operation. Note that the availability of an edge depends on the pre flow; as the pre flow is changed a given edge *a b* may gain or lose this status. The maximum amount that can be pushed along an available edge *a b* is $\bar{c}_{ab} = c_{ab} - f_{ab} + f_{ba}$. The key to controlling the pushing of flow is to define an available edge a b to be *admissible* if and only if $\lambda(a) = \lambda(b) + 1$. The pre flow push algorithm allows pushes only on admissible edges.

The Pre flow Push algorithm is based on the following local routine that operates on the excess flow at a single vertex.

Process (*v*)
1. Start with an active node *v* (so *ex* (*v*) > 0).
2. Pick an admissible edge *v a*.
3. Push *min* $\{ex(v), \bar{c}_{va}\}$ additional flow along *v a*.
4. Repeat Steps 2 and 3 until either ex (v) = 0 or there are no more admissible edges.
5. If *ex* (*v*) > 0 and there are no admissible then re label *v* as follows. Set
   $\lambda(v) = min\ \{\lambda(a) + 1 |\ v\ a$ is an available edge$\}$
   This will produce at least one admissible edge; return to Step 2. Note that since *ex* (*v*) > 0 this excess flow must have arrived on some arcs that are now available for a push.
6. If *ex* (*v*) = 0 then *v* is no longer active and we are finished processing *v*.

Pre-flow Push Algorithm
1. Initialize: define an initial pre-flow and Distance labeling on *N*
2. While an active node *v* remains: Process (*v*).
3. The pre-flow is now a maximum flow.

The above algorithm can be applied to find the maximum in flow pushed through any flow network. Consider the following network for finding the maximum flow applied.

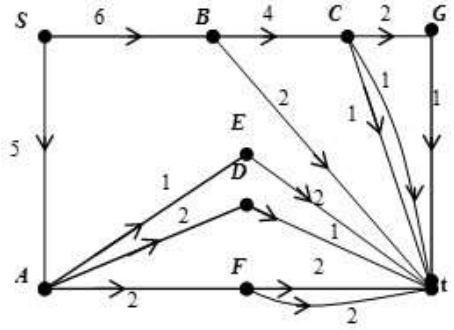
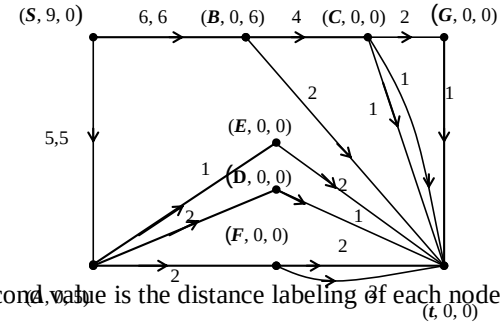


Figure: 3. 1 A simple network

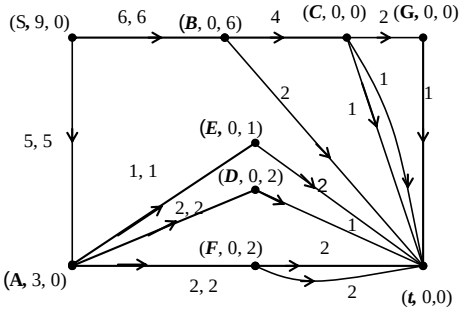
ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

Step 1

1. Initialize the pre-flow and labeling

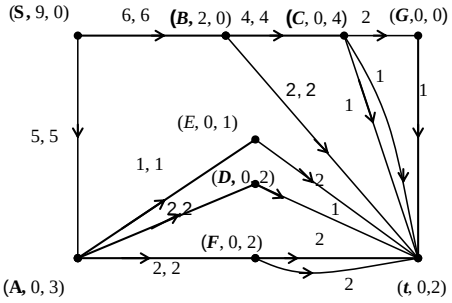


In the above figure, the first variable represent the node, the second value is the distance labeling of each node and the third value is the excess flow of that node.



2. Re label A: $\lambda(A) = 1$
3. Push on AE: $P_{AE} = 1$, $ex\ (A) = 4$
4. Re label A: $\lambda(A) = 2$
5. Push on AD: $P_{AD} = 2$, $ex\ (A) = 2$
6. Re label A: $\lambda(A) = 3$
7. Push on AF: $P_{AF} = 2$, $ex\ (A) = 0$.

Step 2



8. Re label B: $\lambda(B) = 1$
9. Push on BC: $P_{BC} = 4$, $ex\ (B) = 2$
10. Re label B: $\lambda(B) = 2$
11. Push on $B\,t$: $P_{B\,t} = 4$, $ex\ (B) = 0$.

Step 3

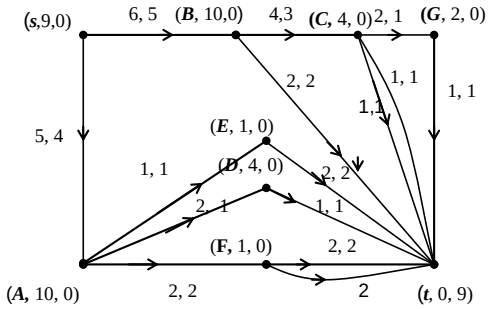
12. Re label E: $\lambda(E) = 1$
13. Push on $E\,t$: $P_{E\,t} = 1$, $ex\ (E) = 0$.
14. Re label D: $\lambda(D) = 1$
15. Push on $D\,t$: $P_{D\,t} = 1$, $ex\ (D) = 1$
16. Re label D: $\lambda(D) = 4$
17. Push on DA: $P_{DA}=1$, $ex\ (D) = 0$.

Step 4

18. Re label A: $\lambda(A) = 10$
19. Push on AS: $P_{SA} = 1$, $ex\ (A) = 0$.
20. Re label F: $\lambda(F) = 1$
21. Push on Ft: $P_{F\,t} = 1$, $ex\ (F) = 0$.
22. Re label C: $\lambda(C) = 1$
23. Push on CG: $P_{CG} = 1$, $ex\ (C) = 2$
24. Re label C: $\lambda(C) = 2$
25. Push on Ct: $P_{C\,t} = 1$, $ex\ (C) = 1$.
26. Re label C: $\lambda(C) = 3$
27. Push on Ct: $P_{C\,t} = 1$, $ex\ (C) = 0$.

ISSN: 2277-9655
(I2OR), Publication Impact Factor: 3.785

28. Re label G: $\lambda(G) = 1$
29. Push on G t: $P_{G\,t} = 1$, $ex\,(G) = 1$
30. Re label G: $\lambda(G) = 2$
31. Push on CG: $P_{CG} = 1$, $ex\,(G) = 0$.
32. Re label C: $\lambda(C) = 4$
33. Push on CB: $P_{CB} = 1$, $ex\,(B) = 1$
34. Re label B: $\lambda(B) = 10$
35. Push on BS: $P_{BS} = 1$, $ex\,(B) = 0$.
36. Stop: No active vertices remain
    ∴ Maximum in flow = 9.

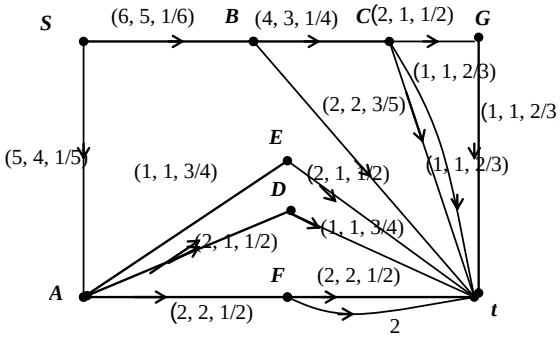


The algorithm sends flow along all gain augmenting paths simultaneously, using a maximum flow computation. The algorithm is given below.

Algorithm 3.1:
Procedure max out flow (X: network; f: flow; Gain function $\gamma : E \to R$: max flow: real)
{Finds maximum 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



In this network we have GAP:
F_1: $S - B - t$, $S - B - C$ - t, is increased by one in 2 ways and $S - B - C - G$ - t,
F_2: $S - A - E$ - t, $S - A - D$ - t and $S - A - F - t$.
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 $S - B$ - t can be expressed as

$$x \to \frac{5x}{6} = 0.833x, \quad \frac{5x}{6} \to \frac{5x}{6} * \frac{2}{5} = 0.833x * 0.4$$

In the path $S - B - C$ - t, the expression becomes

$$x \to \frac{5x}{6} = 0.833x, \quad \frac{5x}{6} \to \frac{5x}{6} * \frac{3}{4} * \left(\frac{1}{3} + \frac{1}{3}\right) = 0.833x * \frac{3}{4} * \frac{2}{3} = 0.833x * 0.5$$

In the path *S* - *B* – *C* - *G* – *t*, the expression can be written as

$$x \to \frac{5x}{6} = 0.833x, \quad \frac{5x}{6} \to \frac{5x}{6} * \frac{3}{4} * \frac{1}{2} * \frac{1}{3} = 0.833x * 0.125$$

In F_2, The gain function of the *S* – *A* – *E* - *t* can be expressed

$$y \to \frac{4y}{5} = 0.8y, \quad \frac{4y}{5} \to \frac{4y}{5} * \frac{1}{4} * \frac{1}{2} = 0.8y * 0.125$$

In the path *S* – *A* – *D* - *t*, the expression can be written as

$$y \to \frac{4y}{5} = 0.8y, \quad \frac{4y}{5} \to \frac{4y}{5} * \frac{1}{2} * \frac{1}{4} = 0.8y * 0.125$$

In the path *S* – *A* – *F* - *t*, the expression can be written as

$$y \to \frac{4y}{5} = 0.8y, \quad \frac{4y}{5} \to \frac{4y}{5} * \frac{1}{2} * \frac{1}{2} = 0.8y * 0.25$$

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 (*S*, *B*, *t*)

The gain function $\frac{5x}{6} = 0.833x$

0.833*x* * 0.4 = 0.332

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

0.833*x* * 0.5 = 0.4166

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

0.833*x* * 0.125 = 0.1041

When *y* = 1, Augment flow along (*S*, *A*, *E*, *t*)

The gain function $\frac{4y}{5} = 0.8y$

0.8*y* * 0.125 = 0.1

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

0.8*y* * 0.125 = 0.1

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

0.8*y* * 0.25 = 0.2

Maximum out flow = 0.332 + 0.4166 + 0.1041 + 0.1 + 0.1+ 0.2 = 1.2539.

Iteration 2:

When *x* = 2, Augment flow along (*S*, *B*, *t*)

The gain function $\frac{5x}{6} = 0.833x = 0.833 * 2 = 1.666$

1.6666*x* * 0.4 = 0.6666

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

1.6666*x* * 0.5 = 0.8333

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

1.6666*x* * 0.125 = 0.2083

When *y* = 2, Augment flow along (*S*, *A*, *E*, *t*)

The gain function $\frac{4y}{5} = 0.8y = 0.8 * 2 = 1.6$

1.6*y* * 0.125 = 0.2

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

1.6y * 0.125 = 0.2
Augment flow along (S, A, F, t)
1.6y * 0.25 = 0.4
Maximum out flow = 0.6666 + 0.8333 + 0.2083 + 0.2 + 0.2 + 0.4 = 2.5082
By iteration 3, Maximum out flow = 3.7619.
Proceeding like this, we will increase the value of x and y, up to x = 6 and y = 5, since the flow value cannot exceed the capacity value.

Result:
Maximum in flow = 9, Maximum out flow = 7.1247.

The algorithm stops in a finite number of steps with a maximum flow and we can estimate the number of steps used. There are several facts to be checked. First, we check that the algorithm maintains both a valid feasible pre flow and a valid distance labeling at each stage. If there are n nodes, then the algorithm takes at most $2n^2$ relabel steps and at most $3n^3$ pushes. Also we know that the algorithm doesn't stop while there is an active node. We can show that if a feasible flow F has a distance labeling the F is a maximum flow. It ensures that when the algorithm does stop it has determined a maximum flow.

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 modeled using generalized flows. Later, we will use these problems 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. Ohlin, Network Flow, Prentice Hall, Englewood Cliffs, N. J. (1993).
[3] F. Glover, J.Hultz, D. Klingman, and J. Stutz. Generalized networks: A fundamental computer based planning tool. Management Science, 24:1209-1220, (1978).
[4] A. V. Goldberg, R. E. Trajan, A New Approach to the Maximum Flow Problem, J. ACM 35 (1988) 921-940.
[5] H. S. Wolf, Algorithms and Complexity, Prentice Hall International, Inc., U. S. A. (1986).