Development of An Algorithm for All Type of Network Flow Problems

Pawan Tamta
Department of Mathematics
S.S.J. Campus Almora
Kumaun University, Nainital
Uttarakhand, India

B. P. Pande
Department of IT
S.S.J. Campus Almora
Kumaun University, Nainital
Uttarakhand, India

H. S. Dhami
Department of Mathematics
S.S.J. Campus Almora
Kumaun University, Nainital
Uttarakhand, India

ABSTRACT
We develop an algorithm which reduces the arbitrary instance of the network flow problem to a simple path disjoint network in polynomial time. Then the flow in each path is taken as the minimum of the arc capacities of that path from where the flow in each arc can be easily determined. The polynomial time algorithm can determine any instance of the network flow problem faster than the previously existing algorithms . An example has been given to elucidate the process. At the end a MATLAB program based on this algorithm has been given.

General Terms
Network flow problem, computational complexity

Keywords
Maximum Flow Network Problem (MFNP), Simple path disjoint network, polynomial time algorithm.

1. INTRODUCTION
In the Maximum Flow Network Problem (MFNP), we are given a capacitated *s–t* (directed) network where each arc has a fixed capacity. The objective is to determine the flow in each arc and the maximum flow that can be routed through the network.

Network flow problems model the essential issues in many real-world resource allocation problems, including coordinating tactical air strikes[13], combating drug trafficking [19], controlling infections in a hospital [2], chemically treating raw sewage [14], and controlling floods [15] The study of network flow problem is basic to many advanced problems such as Maximum Flow Network Interdiction Problem(MFNIP) in particular originates from the Cold War, when analysts at the RAND Corporation studied how to interdict the Soviet Union's railroad traffic into Eastern Europe using the fewest resources [9], They accomplished this by solving a Minimum Capacity *s–t* Cut Problem; this is the earliest known formulation of this fundamental problem [17]. From the 1960s to the turn of the 21st century, there has been an extensive amount of academic literature on various network problems. All the models aimed at finding the solution by considering the arc capacities and less attention has been given to the simplification of the network, most of which is listed in [6]. The basic framework for maximum flow network interdiction with explicit budget constraints was first studied in [13].In the early 1990s, Wood [19], resurrected interest in network interdiction by introducing MFNIP. A similar problem to MFNIP, called the Network Inhibition Problem (NIP), was independently introduced by Phillips in [14]. Numerous variants of network interdiction problems have also been studied, including shortest path network interdiction [10], stochastic network interdiction [7] and [11], multiple commodity network interdiction [12] and[19], facility interdiction[6], and a variant of MFNIP where flow is routed before arcs are removed [8]. There is also literature on more-than-two-stage interdiction models where infrastructure may be reinforced against attacks [4]. The special case of MFNIP when an interdictor removes exactly *k* arcs from the network in order to minimize the maximum flow in the resulting network is known as the *k*-Most Vital Arcs Problem [15]; in [19], this problem is called the Cardinality Maximum Flow Network Interdiction Problem (CMFNIP).

In this paper we develop an algorithm which reduces any arbitrary instance of the network problem to a simple source to node path disjoint network in polynomial time. The maximum flow in the network, the flow in each arc and the cuts of the network can be determined in a simple way as compared to the previously existing methods in which we have to determine the residuals in each step and then after a lengthy calculation we can determine the flow in each arc.

2. PRELIMINARIES
We denote a network as (N, A) where N is the set of nodes and A is the set of arcs. We assume without loss of generality that all of our networks have a unique source $s \in N$ and a unique sink $t \in N$. An arc that originates from node u and terminates at node v is denoted as (u, v). For a node v we denote the set of all arcs entering node v as $\delta^+(v)$, and we denote the set of all arcs leaving node v as $\delta^-(v)$. We refer to an *s–t* cut as either a set of arcs that disconnects s from t upon their removal, or alternatively, as a bipartition of the nodes where s and t are not in the same partition. Since the only cuts of interest in this paper are *s–t* cuts, we henceforth refer to them as cuts. Similarly, we denote an undirected graph as (V, E) where V is the set of vertices and E is the set of edges. We denote an edge between vertices u and v by $\{u,v\}$.And an arc between node i and j as (i, j). The arc capacity of every arc (i, j) is given as C_{ij}. The set of different paths from node i is given as P_i^t and a particular path is given as p_i^t. Each source to sink path is given as Pn_i. The flow in path Pn_i is given as Fn_i and flow in arc (i,j) as $Fn_{(i,j)}$. Nodes which are directly connected to source node are of main concern here so we define this node set as A.Obviously $A \subset N$.

3. SOURCE TO NODE DISJOINT PATH
Here we give a function which defines source to node disjoint path. The special feature of the path is that two consecutive

arcs are not common in any two paths. As a starting node here we consider the node set which is directly connected to the source node without any intermediate node.

$Pn_i = (s, i) + p_i^t$ (3.1)

s.t $\forall\ p_i^t \in P_i^t$

and $\forall\ (l,m) + (m,n) \in p_i^t$

$\Rightarrow (l.m) + (m,n) \notin p_j^t\ \forall\ j \neq i$ (3.2)

Flow in Pn_i

$Fn_i = Min(C_{ij})$ (3.3)

if $(l,m) \in Pn_i$ and $(l,m) \notin Pn_j\ \forall\ j \neq i$

then flow in arc $(l,m) = Fn_i$ (3.4)

if $(l,m) \in \cap_i Pn_i$

Then flow in arc $(l,m) = \Sigma_i Fn_i$ (3.5)

Maximum flow in the network $= \Sigma_i Fn_i$, $\forall\ i \in A$ (3.6)

Equation 3.1 determines source to sink path for every node $n \in A$, equation 3.2 imposes the condition that any two paths can not have two arcs common if they are in sequence. Therefore the possibility of the portion of any path having more than one arc, is being restricted here.

Equation 3.3 gives the flow Fn_i in every path Pn_i which is minimum arc capacity of the arcs constituting the path Pn_i .

Equation 3.4 determines the flow in any arc belonging to path Pn_i which is same as the flow in path Pn_j if that arc uniquely belongs to Pn_i.

Equation 3.5 determines the flow in any arc (l,m) if it belongs to more than one path as sum of the flow in respective paths.

Equation 3.6 determines the maximum flow in the network as sum of the flows in every arc.

Equations 3.1 to 3.6 altogether constitutes the algorithm which gives the maximum flow in the network, flow in each arc and determines the cut.

4. ALGORITHM TRANSFORMS THE COMPLICATED NETWORK IN TO A SIMPLE SOURCE TO SINK PATH NETWORK IN POLYNOMIAL TIME

The equation set 3.1 and 3.2 constitute the algorithm which transforms the network in polynomial time. In any directed graph having N nodes the maximum number of arcs are N(N-1).The algorithm searches each arc for the possible source to sink path. Therefore the maximum number of efforts can not exceed $N(N-1) < N.N = N^2$.Efforts being bounded above by the polynomial of degree 2 shows that equation set 3.1 and 3.2 is a polynomial time algorithm of degree 2.

The algorithm can solve arbitrary instance of the network flow model efficiently.

Here we consider the network (fig 4.1) with given arc capacities. After each source to sink path determined by equation 3.1 and 3.2 the network easily gets transformed in to (fig 4.2). Then the flow in each path and every arc is determined by equations 3.3 to3.6

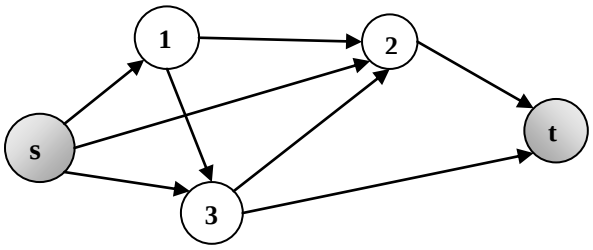

Fig 4.1: Initial Network

Arc capacities are given as

$C_{s1} = 20$, $C_{12} = 10$, $C_{2t} = 8$, $C_{13} = 12$, $C_{s2} = 15$, $C_{32} = 8$, $C_{s3} = 13$, $C_{3t} = 9$

By the application of the above mentioned function this network can be reduced as Figure 4.2

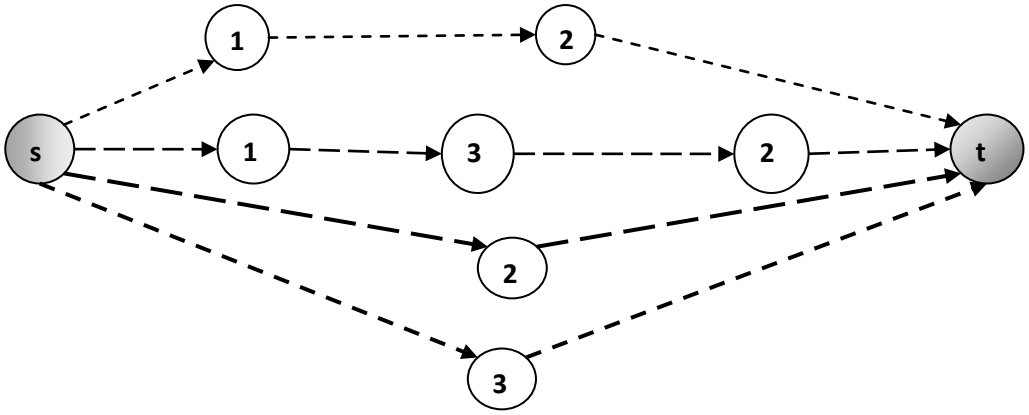

Fig 4.2: Network after reduction

5. RESULTS

Referred to figure 4.2, Flow in different path is given as

$F_1 = 8$, $F_2 = 8$, $F_3 = 8$, $F_4 = 9$

Arc capacities are given as

$C_{s1} = 8+8 = 16$, $C_{12} = 8$, $C_{2t} = 8+8+8 = 24$, $C_{13} = C_{32} = 8$, $C_{s2} = 8$, $C_{s3} = 9$, $C_{3t} = 9$

Maximum flow in the network, $F_1 + F_2 + F_3 + F_4 = 33$.

6. CONCLUSION

The complicated network has been reduced to the simplest source to node path network without any intermediate node, which can be applied to any complicated network related problems such as network flow interdiction problem. The algorithm has the capability to reduced the network in polynomial time from where the flow in each arc, cuts and total flow has been easily calculated.

7. REFERENCES

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin, Network Flows: Theory, Algorithms and Applications, Prentice Hall, Upper Saddle River (1993).

[2] N. Assimakopoulos, A network interdiction model for hospital infection control, *Computers in Biology and Medicine* **17** (1987), pp. 413–422

[3] L. Bingol, A Lagrangian heuristic for solving a network interdiction problem, Master's thesis, Naval Postgraduate School, 2001.

[4] G.G. Brown, M.W. Carlyle, J. Salmerón and R.K. Wood, Defending critical infrastructure, *Interfaces* **36** (2006), pp. 530–544

[5] C. Burch, R.D. Carr, M. Marathe, C.A. Phillips and E. Sundberg, A decomposition-based pseudoapproximation algorithm for network flow inhibition. In: D.L. Woodruff, Editor, *Network Interdiction and Stochastic Integer Programming*, Kluwer Academic Press (2002), pp. 51–68.

[6] R.L. Church, M.P. Scaparra and R.S. Middleton, Identifying critical infrastructure: The median and covering facility interdiction problems, *Annals of the Association of American Geographers* **94** (2004), pp. 491–502. |

[7] K.J. Cormican, D.P. Morton and R.K. Wood, Stochastic network interdiction, *Operations Research* **46** (1998), pp. 184–197.

[8] D. Du and R. Chandrasekaran, The maximum residual flow problem: NP-hardness with two-arc destruction, *Networks* **50** (2007), pp. 181–182.

[9] T.E. Harris, F.S. Ross, Fundamentals of a method for evaluating rail network capacities, Research Memorandum RM-1573, The RAND Corporation, Santa Monica, CA.

[10] E. Israeli and R.K. Wood, Shortest-path network interdiction, *Networks* **40** (2002), pp. 97–111

[11] U. Janjarassuk and J.T. Linderoth, Reformulation and sampling to solve a stochastic network interdiction problem, *Networks* **52** (2008), pp. 120–132.

[12] C. Lim and J.C. Smith, Algorithms for discrete and continuous multicommodity flow network interdiction problems, *IIE Transactions* **39** (2007), pp. 15–26.

[13] A.W. McMasters and T.M. Mustin, Optimal interdiction of a supply network, *Naval Research Logistics Quarterly* **17** (1970), pp. 261–268.

[14] C.A. Phillips, The network inhibition problem, in: Proceedings of the 25th Annual ACM Symposium on the Theory of Computing, 1993 pp. 776–785.

[15] H.D. Ratliff, G.T. Sicilia and S.H. Lubore, Finding the *n* most vital links in flow networks, *Management Science* **21** (1975), pp. 531–539.

[16] J.O. Royset and R.K. Wood, Solving the bi-objective maximum-flow network-interdiction problem, *INFORMS Journal on Computing* **19** (2007), pp. 175–184.

[17] A. Schrijver, On the history of the transportation and the maximum flow problems, *Mathematical Programming* **91** (2002), pp. 437–445

[18] A. Uygun, Network interdiction by Lagrangian relaxation and branch-and- bound, Master's thesis, Naval Postgraduate School, 2002.

[19] R.K. Wood, Deterministic network interdiction, *Mathematical and Computer Modelling* **17** (1993), pp. 1–18.

8. APPENDIX

MATLAB Program of the algorithm

```
>> A=[1,3];

>> Cs1=20;

>> Cs3=13;

>> C12=10;

>> Cs2=15;

>> C13=12;

>> C32=8;

>> C2t=8;

>> C3t=9;

>> P1=[Cs1,C12,C2t];

>> P2=[Cs1,C13,C32,C2t];

>> P3=[Cs2,C2t];

>> P4=[Cs3,C3t];

>> F1=min(P1)

F1=8

>> F2=min(P2)

F2 = 8

>> F3=min(P3)

F3 = 8
```

```
>> F4=min(P4)

F4 = 9

>> Fs1=F1+F2

Fs1 = 16

>> Fs2=F3

Fs2 = 8

>> Fs3=F4

Fs3 = 9

>> F12=F1

F12 = 8

>> F2t=F1+F2+F3

F2t = 24

>> F13=F2

F13 = 8

>> F32=F2

F32 = 8

>> F3t=F4

F3t = 9

>> Mf=F1+F2+F3+F4

Mf = 33
```