PES University, Bangalore

(Established under Karnataka Act No. 16 of 2013)

UE21MA141B- LINEAR ALGEBRA AND ITS APPLICATIONS

LAA- PROJECT

Session: Jan-May 2023

Branch : CSE

Semester & Section : 4 E

Sl No.	Name of the Student	SRN	Marks Allotted (Out of 5)
1.	Maryam Khan	PES2UG21CS283	
2.	Melvin A Gomez	PES2UG21CS293	
3.	Mudundi Manasvi Varma	PES2UG21CS305	
4.	Ananya Bhatia	PES2UG21CS903	

Name of the Course Instructor : Dr. Girish V R

Signature of the Course Instructor

(with Date) : _____

Linear Algebra (UE21MA241) - PROJECT REPORT

FLOW IN NETWORKS

With SciLab

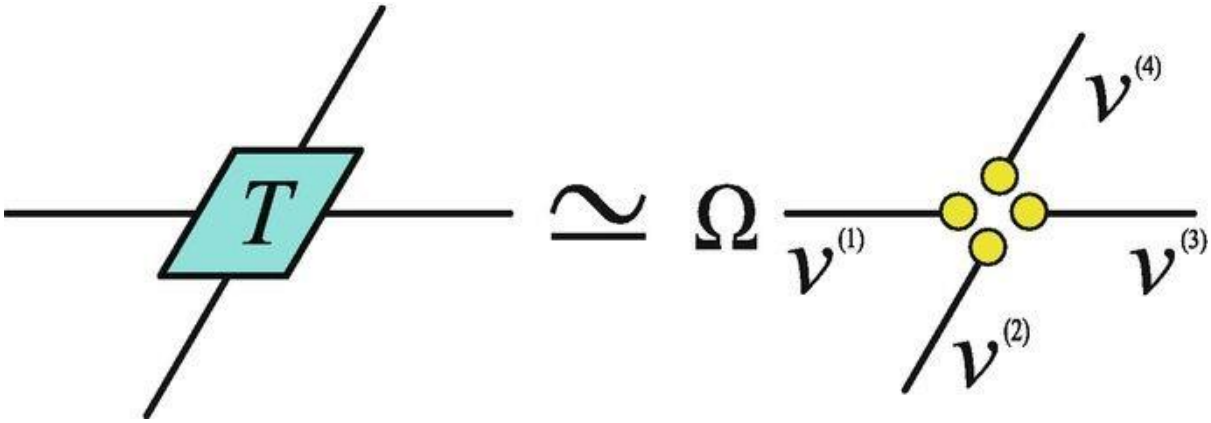


Table Of Contents

Abstract

"Mathematics rightly viewed possesses not only the truth but supreme beauty"

-Bertrand Russell

This project explores the fundamental concept of flow in networks, which has wide-ranging applications in transportation, communication, and energy systems. Network flow problems are commonly represented by graphs, and abstract linear algebra provides a powerful framework for analyzing flow through the edges of the graph. By representing flows as vectors and using linear transformations to model flow through a network, complex flow problems can be analyzed and optimized.

This report demonstrates how linear programming can be applied to solve network flow problems, which have significant real-world implications. By formulating network flow problems as linear programs, a polyhedral set of feasible solutions can be obtained. This report highlights the similarities and differences between network flow problems and linear algebra programs, and showcases how network flow problems can be interpreted and solved using linear programming.

Introduction

Network Flow

Flow in networks is an important concept in network theory, and it has applications in a variety of fields such as transportation, communication, and energy systems. To study flow in networks, abstract linear algebra is often used as a mathematical tool. The notion of a vector space is a key concept in linear algebra, and in the context of network theory, vectors can be used to represent flows through the edges of a graph.

In addition to vectors, linear algebra provides a way to represent linear transformations, which can be used to model the flow of a quantity through a network. For example, traffic flow in a transportation network can be modeled as a vector, where each entry corresponds to the amount of traffic flowing through a particular road. The flow can be represented as a linear transformation, which takes the traffic vector as input and produces a new vector representing the flow after passing through the network.

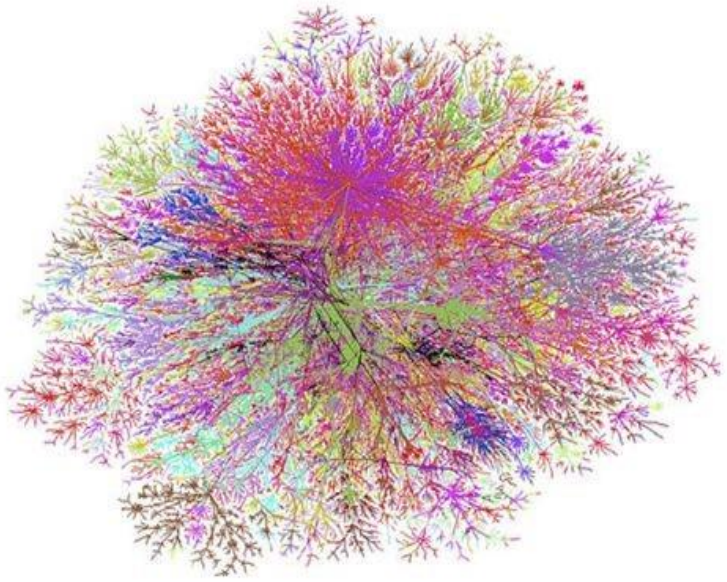
Network flow analysis has applications in a wide range of fields, including transportation, logistics, telecommunications, and computer networking. The goal of network flow analysis is to determine the optimal flow through the network, subject to constraints such as the capacity of the edges and the demand at each node. This can be done using algorithms such as the max-flow min-cut algorithm or the shortest path algorithm.

One of the challenges in network flow analysis is determining the optimal flow through the network. The max-flow min-cut algorithm is a widely used algorithm for finding the optimal flow. It works by finding the minimum capacity of a cut that separates the source node from the sink node. The max-flow through the network is equal to the min-cut capacity.

In addition to the max-flow min-cut algorithm, other algorithms such as the shortest path algorithm can be used to find the optimal flow through the network. The shortest path algorithm works by finding the shortest path between the source node and the sink node. The optimal flow is then determined based on the capacity of the edges along the shortest path.

In conclusion, flow in networks is an important concept in network theory and has numerous applications in real-world systems. Abstract linear algebra provides a powerful tool for analyzing flow in networks, and algorithms such as the max-flow min-cut algorithm and the shortest path algorithm can be used to determine the optimal flow through the network. Further research in this area can lead to the development of more efficient algorithms for network flow analysis, with potential applications in various fields such as transportation, logistics, telecommunications, and computer networking.

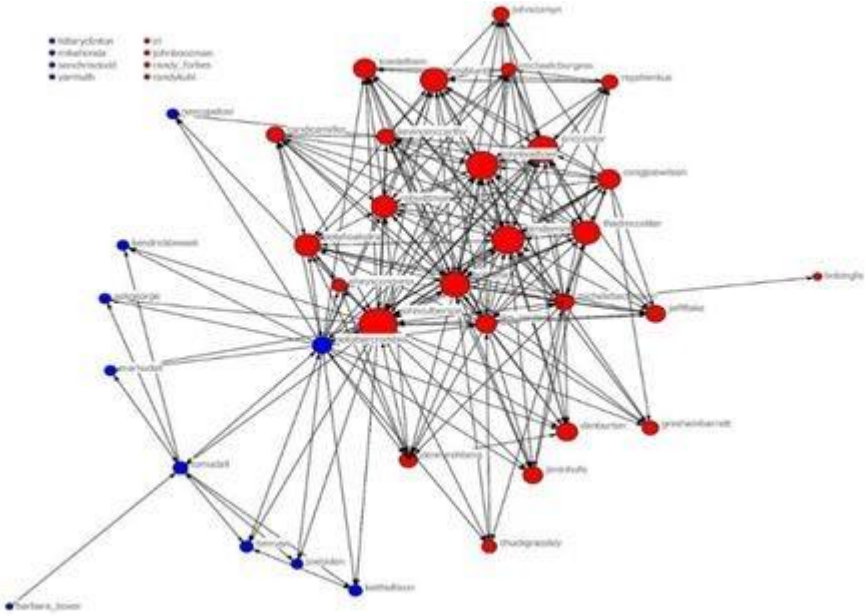
A Few Examples of Network Flow are:



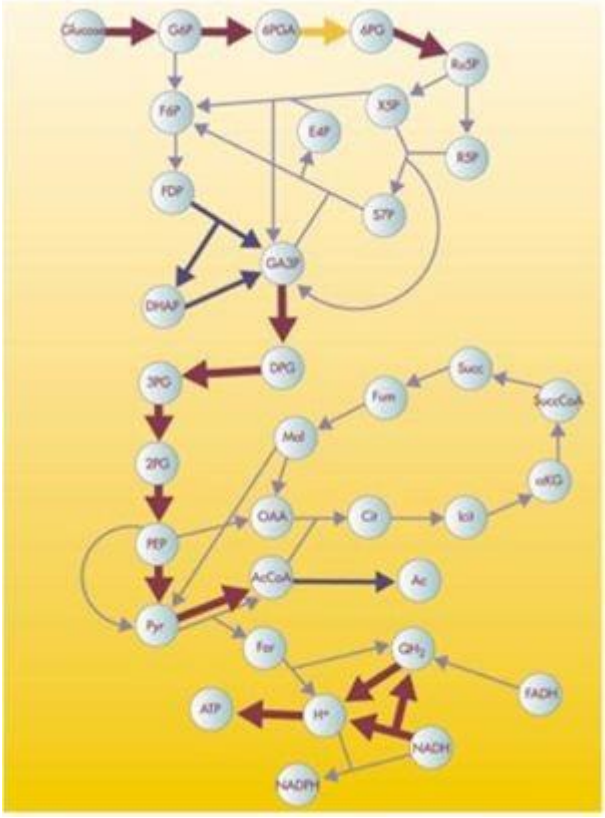
The image shows Network of connections between devices within the Internet. Colours indicate operator of network. Structure determined by sending a storm of IP packets out randomly across the network. Each packet is programmed to self-destruct after a delay, and when this happens, the packet failure notice reports back the path the packet took before it died.

The World Wide Web is a huge network where the pages are nodes and links are the edges.

Other examples include social networks of acquaintances or other types of interactions, networks of publications linked by citations, transportation networks, metabolic networks, and communication networks.



A map of the US congressmen and women who were on Twitter in January 2009. The direction of the arrows show who follows whom, and the size of the blobs indicates how popular a given congressperson is among their twittering peers (where popularity indicates the number of peer followers). Colours indicate party affiliation (blue dots are Democrats, red dots are Republicans).



Schematic of the metabolic pathways in E. Coli. Obtained from Wikimedia Commons, which in turn obtained the image from http://genomics.energy.gov website i.e. The U.S. Department of Energy Genome Programs. The image is in the public domain.

Mathematical definition of Network and Network Flow:

A network is a graph G = (V, E), where V is a set of vertices and E is a set of V's edges (a subset of VxV), together with a non-negative function

$c: V \times V \to \mathbb{R}_\infty$ called capacity function.

We may assume that if (u, v) ∈ E then (v, u) is also a member of E, since if (v, u) ∉E then we may add (v, u) to E and then each set c(v, u) = 0, without loss of generality. If two nodes in G are distinguished, a source and 's' and a sink 't', then (G, c, s, t) is called a flow network (here c is a capacity function which tells the capacity associated with each edge in the graph).

Problem Statement

Problem Statement

By increasing the complexity of networks which we interact with in our day to day life, it is important for us to understand the structure of those networks.

Some common questions that pop up in our heads are:

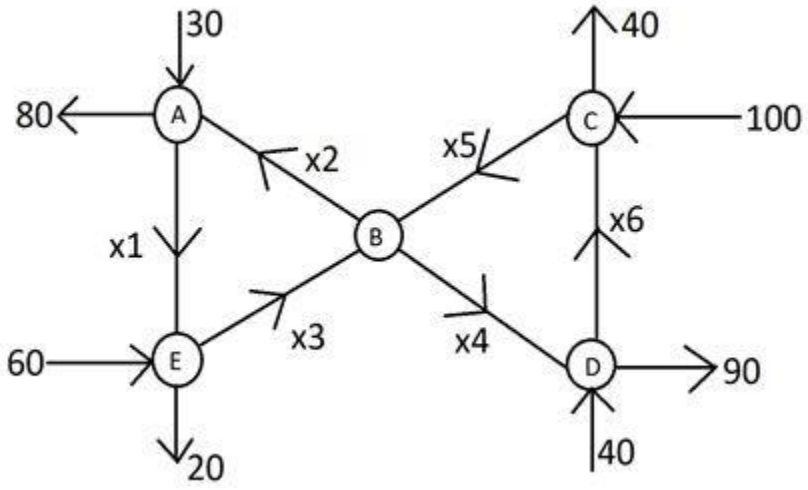
1. How the flow will occur I.e., analysing the flow
2. How to find the flow between any two nodes given to us
3. Is there any path possible between given two nodes
4. Finding the set of nodes that are connected to each other
5. Findings the cyclic structures given any complicated graph
6. Finding and the nodes which can act like sink nodes
7. Commenting on the capacity of the any given edge in network

The answer to all such problems are answered in this project

Linear Algebra in Network Flow

Let us begin with simple things we often do with all types of Network i.e. analyzing the flow patterns.

Finding the flow inside the network and calculating values x1 to x6, give us a small glimpse of how Linear algebra can be used to make things easy.



In all the networks, we know that in-flow must be equal to out-flow following the conservation constraint.

This will lead us to 5 different equations which are :

At A ⇒ 30 + x2 = 80 + x1

At B ⇒ x3 + x5 = x2 + x4

At C ⇒ 100 + x6 = 40 + x5

At D ⇒ 40 + x4 = 90 + x6

At E ⇒ x1 + 60 = x3 + 20

Solving the above equations will give us the values of the flow inside the network.

But how can we solve large numbers of linear equations easily?

By applying Gauss-Jordan Method of Elimination

The Augmented Matrix representation of the above equations would look like

$$\left[\begin{array}{cccccc|c} -1 & 1 & 0 & 0 & 0 & 0 & 50 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & -60 \\ 0 & 0 & 0 & 1 & 0 & -1 & 50 \\ 1 & 0 & -1 & 0 & 0 & 0 & -40 \end{array}\right]$$

A b

On reducing the matrix we get

$$\left[\begin{array}{cccccc|c} 1 & 0 & -1 & 0 & 0 & 0 & -40 \\ 0 & 1 & -1 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 1 & 0 & -1 & 50 \\ 0 & 0 & 0 & 0 & 1 & -1 & 60 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

We get infinite number of solutions with following values

x1 = -40 + x3

x2 = 10 + x3

x3 = x3

x4 = 50 + x6

x5 = 60 + x6

x6 = x6

(where x3 and x6 are free variables)

As the direction of the flow cannot be altered, all the values in the flow should be non negative.

These will lead us to the minimum flow in each of the edge i.e.,

x1 >= 0 implies x3 >= 40 x6 >= 0

The other values would definitely be positive.

Thus we get the minimum flow in each of the edge as

x1 = 0

x2 = 50(10+40)

x3 = 40

x4 = 50(50+0)

x5 = 60(60+0)

x6 = 0

The point of finding the minimum helps us in the practical world where we are bound by limitations. If the above graph represents the network flow of water, finding the minimum value we will give us a rough idea about the capacity of the pipe that should be placed along the edge.

From the above example we have come to the conclusion that for a given flow at the corners of the network, we can calculate the flow inside the network and also talk about the capacity constraints on each of the edges.

We can see that the above graph is connected i.e. From every node there is at least one path to all other nodes.

But practically we need to find connectivity. If we wish to find the nodes that are connected, we need to find the connected component of the largest network.

How do we achieve this?

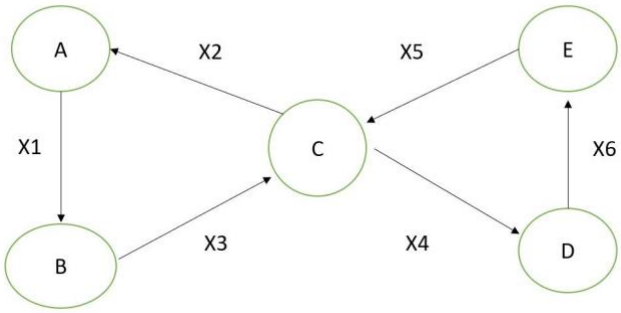
This is where the concept of Fundamental Subspaces comes into picture.

If we focus only on the structure of the graph i.e. Directions of the flow, and apply our conservation constraint where in-flow = out-flow at each node will give us the equation.

Representing those equations in the matrix form and finding the basis of the left null space will give us the connected parts of the graph.

Explanation with example:

Using the previous example with directions



Applying in-flow = out-flow will give us the following equations

$$\text{At A} \Rightarrow x1 - x2 = 0$$

$$\text{At B} \Rightarrow x1 - x3 = 0$$

$$\text{At C} \Rightarrow -x2 + x3 - x4 + x5 = 0$$

$$\text{At D} \Rightarrow x4 - x6 = 0$$

$$\text{At E} \Rightarrow x5 - x6 = 0$$

Above equations when represented in the matrix form

$$\left[\begin{array}{cccccc|c} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{array}\right]$$

Left Null Space = Null space of Transpose of A

Transpose of the above matrix =

$$\left[\begin{array}{rrrrr|r} 1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 0 \end{array}\right]$$

Row reducing above matrix:

$$\left[\begin{array}{rrrrr|r} 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

Required Left Null Space =

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \end{bmatrix} = E \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

Basis of the left Null space =

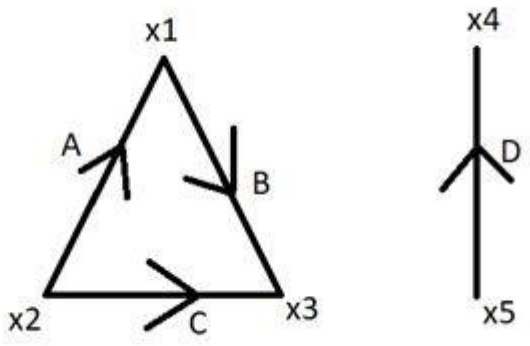
$$\left\{ \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \\ 1 \end{bmatrix} \right\}$$

As expected, we got only one element in the basis of the Left Null Space because our graph is connected.

What if the network is not connected and how can we find the group of connected nodes?

Lets try the above method with a different example

x1, x2, x3 are nodes
A, B, C, D are edges



Apply in-flow = out-flow at each nodes will give us following equations

$$\text{At } x1 \Rightarrow A - B = 0$$

$$\text{At } x2 \Rightarrow A + C = 0$$

$$\Rightarrow$$

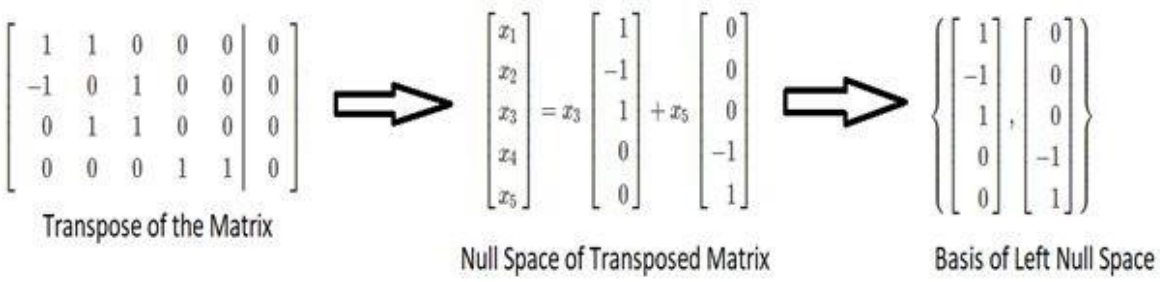
$$\text{At } x3 \Rightarrow B + C = 0$$

$$\text{At } x4 \Rightarrow D = 0$$

$$\text{At } x5 \Rightarrow D = 0$$

Matrix representation of above equations

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$



Now we see that basis has two elements. Which implies that the network has two connected components. And the nodes of connected components is nothing but the non-zero entries of each of the basis i.e. x1, x2, x3 are one connected part and x4, x5 is the other connected part of the network.

The above method is highly appreciated because of its uses in the practical world where networks have thousands of nodes and millions of edges.
It would be very challenging to find the connected components without linear algebra.

So far we talked about 2 key things about a network

1. Analysing the inner flow given outer flow values
2. Finding all the connected nodes of the network.

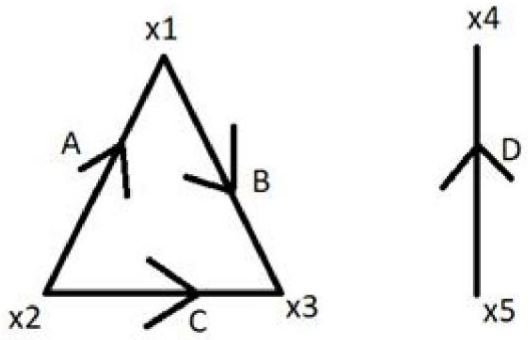
Now let's talk about redundancy. It is useless to have a cyclic structure inside the structure of the network.

How can we find all the cyclic paths in the given network?
Again the Fundamental Subspaces of Linear algebra comes in picture.

Finding the Null space of the matrix representation of the equations formed by the graph showing the structure of the network will give us the Cyclic Paths. Explanation with Example:

Let's take our previous graph again

x1, x2, x3 are nodes
A, B, C, D are edges



Matrix Representation:

$$\left[\begin{array}{cccc|c} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{array}\right]$$

Null Space of the above matrix =

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = C \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \end{bmatrix}$$

Basis of the Null space =

$$\left\{ \begin{bmatrix} -1 \\ -1 \\ 1 \\ 0 \end{bmatrix} \right\}$$

As we know that there is only one independent cyclic path i.e.
from AB Opposite directions of C

If any entry is zero in the null Space (in our case we can see that D = 0 always) it tells us that the edge is pointing to the sink nodes (Nodes which have only in flow but not the out flow).

If the basis of the Null space contains all positive or all negative entries then it means that the cyclic path is formed along with the direction of flow.

Based on the structure of the network, if we wish to remove such paths, it's possible.

Another key factor in analyzing networks is calculating the maximum flow in a flow network (i.e. from source node to sink node). This can be achieved by using the Ford-Fulkerson method.

Linear programming formulation of Ford-Fulkerson algorithm is given by expressing the maximum flow from a source node s to a sink node t as a linear program. The dual of this program yields a minimum cut value linear program and strong duality establishes the Ford-Fulkerson theorem.

Literature Survey

Paper 1

Matrix Scaling by Network Flow

Provided by Günter Rote and Martin Zachariasen

Pub. July 2006 in the 18th Annual ACM-SIAM Symposium on Discrete Algorithms New Orleans, USA

http://page.mi.fu-berlin.de/rote/Papers/pdf/Matrix+scaling+by+network+flows.pdf

We know that a given nonnegative n × n matrix A = (a_{ij}) can be scaled, by multiplying its rows and columns by unknown positive multipliers λ_i and μ_j , such that the resulting matrix ($a_{ij}\lambda_i\mu_j$) has specified row and column sums r_i and s_j.

This paper talks about an algorithm that achieves the desired row and column sums with a maximum absolute error ε in $O(n^4(\log n + \log (h/\varepsilon))$ steps, where h is the overall total of the result matrix. The algorithm is a scaling algorithm. It solves a sequence of more and more refined discretization. The discretization are minimum cost network flow problems with convex piecewise linear costs. The discretization is interesting in their own right because they arise in proportional elections.

Paper 2

Parallel structured networks for solving a wide variety of matrix algebra problems

Provided by Li-Xin Wang and Jerry M. Mendel

Pub. March 1992 in Journal of Parallel and Distributed Computing

https://www.sciencedirect.com/science/article/abs/pii/074373159290066V

In this paper, structured networks, which are multilayer feed-forward neural networks with linear neurons and with additional constraints on the weights of neurons, are developed for solving a wide variety of matrix algebra problems, including LU decomposition, matrix inversion, QR factorization, singular value decomposition, symmetric eigenproblem, and Schur decomposition.

The basic idea of the structured network approaches is: first, represent a given matrix algebra problem by a structured network so that if the network matches a set of desired patterns, the weights of the linear neurons give the solution to the problem; then, train the structured network to match the desired patterns, and obtain the solution to the problem from the converged weights of neurons.

Modified error back-propagation algorithms are developed for training the structured networks. Time complexities of these structured network approaches are analyzed. These approaches are tested for random matrices through Monte Carlo simulations. Finally, a general-purpose structured network is developed which can be programmed to solve all the matrix algebra problems considered in this paper.

Approach & Solution with Scilab Code

Find the columns that are in the column space of A where
A is [4,5,9,-2; 6,5,1,12; 3,4,8,-3]

```
clear ;

disp('The given matrix is ');
a=[4,5,9,-2; 6,5,1,12; 3,4,8,-3]
a(2,:)=a(2,:)-(a(2,1)/a(1,1))*a(1,:)
a(3,:)=a(3,:)-(a(3,1)/a(1,1))*a(1,:)
disp(a)
a(3,:)=a(3,:)-
(a(3,2)/a(2,2))*a(2,:)
disp(a) a(1,:)=a(1,:)/a(1,1)
a(2,:)=a(2,:)/a(2,2) disp(a)
for i=1:3     for j=i:4
if(a(i,j)<>0)
 disp('is a pivot column',j,'column')
break
end
```

OUTPUT:

The given matrix is

```
   4.   5.     9.    -2.

   0.  -2.5   -12.5   15.

   0.   0.25   1.25  -1.5


   4.   5.    9.    -2.

   0.  -2.5  -12.5   15.

0.   0.    0.     0.
```

1. 1.25 2.25 -0.5

0. 1. 5.

-6. 0. 0.

0. 0.

column

1.

is a pivot column

column

2.

is a pivot column Find the fundamental subspaces of A=[0,1,0; 0,0,1; 0,0,0] clear;

```
A =[0,1,0; 0,0,1; 0,0,0];
disp (A , 'A= ' ) ; [m , n ]=
size (A) ; disp (m , 'm= ' ) ;
disp (n , ' n= ' ) ; [v , pivot
]= rref (A) ; r = length (
pivot ) ; disp (r , ' rank= ' )
cs = A (: , pivot ) ; disp (
cs , ' Column space= ' ) ;
ns = kernel (A) ; disp ( ns ,
' Null space= ' ) ; rs = v (1:
r ,:)'; disp ( rs , 'Row
space= ' ) lns = kernel (A')
; disp ( lns , 'Left
nullspace= ' ) ;
```

OUTPUT

A=

0. 1. 0.

0. 0. 1.

0. 0. 0.

m=

3.

n=

3.

rank=

2.

Column space=

1. 0.

0. 1.

0. 0.

Null space=

1.

0.

0.

Row space=

0. 0.

1. 0.

0. 1.

Left nullspace=

0.

0.

1.

Conclusion

1. Analysis using Gauss Jordan Method of Elimination.
2. Left Null Space to find the connected nodes of Network.
3. Null space to find the Cyclic paths and the sink Nodes of the Network.
4. Solution with aid of SciLab

Bibliography

1. http://ita.ucsd.edu/wiki/index.php?title=Linear_programming_formulation
2. https://www.imsc.res.in/~knr/past/la_to_nw.pdf
3. http://page.mi.fu-berlin.de/rote/Papers/pdf/Matrix+scaling+by+network+flows.pdf
4. https://msp.org/pjm/1957/7-2/pjm-v7-n2-p04-p.pdf
5. https://mathinsight.org/network_introduction