Overview: Graphs & Linear Algebra

Peter M. Kogge
Material based heavily on the Class Book
“Graph Theory with Applications…” by Deo
and
“Graphs in the Language of Linear Algebra: Applications, Software, and Challenges”
Ed. by Jeremy Kepner and John Gilbert[1]

[1] https://www.researchgate.net/profile/Aydin_Buluc/publication/235784365_New_Ideas_in_Sparse_Matrix-Matrix_Multiplication/links/00b495320c1897cddc000000/New-Ideas-in-Sparse-Matrix-Matrix-Multiplication.pdf

Conventional Matrix Operations

Good Tutorial:

https://stattrek.com/matrix-algebra/matrix.aspx

Basic Matrix Operations

- **Pointwise operations**: A, B both NxM
  - If C = A + B, then C[i,j] = A[i,j] + B[i,j]
    - Where + is "natural" scalar addition, And + is matrix addition
    - Written C = A .+ B
  - Same for C = A*B where C[i,j] = A[i,j] * B[i,j]
    - Written C = A .* B
- **Scalar-Matrix operations**: s a scalar, A NxM
  - If C = s + A, C[i,j] = s + A[i,j]
  - Similar for C = s*A (sometimes written sA) or A*s
- **Vector Scaling**: v N elt vector, A NxM
  - If C = v.*A, then C[i,j] = v[i]*A[i,j]

More Basic Matrix Operations

- **Matrix Multiplication**: A is NxM, B is MxR
  - If C = AxB (also written just AB)
  - C[i, k] = A[i,1]*B[1,k] + A[i,2]*B[2,k] + … A[i,N]*B[N,k]
  - Written C = A+.*B
  - Either A or B, or both, could be vectors Nx1, Mx1
- **Matrix Exponentiation**: A NxN
  - If C = A^k, then C = A(A(A…(AA)…) k times
- **Matrix Transpose**: A is NxM
  - If C = A^T, then C is MxN, C[i,j] = A[j,i]

More Basic Matrix Operations

- **Inner Product:** x,y of length N
  - If C = x +.* y, then $C = \Sigma_{i=1,N}$ x[i]*y[i]
  - Also written x•y
- **Outer Product**: x of length N, y length M
  - If C = x ◦ y, then C[i,j] = x[i]*y[j], an NxM matrix
- **Diagonalization**: v a N elt vector
  - If C – diag(v), then C[i,i] = v[i]; C[i,j] = 0, i!=j

Matrix Operation Properties

- If A, B, matrices of same dimensions
  - A + B = B + A (elt-by-elt addition is **commutative**)
  - A + (B + C) = (A + B) + C (also **associative**)
  - Likewise for elt by elt multiplication
- If A is NxM, B is MxR, C is RxQ:
  - A(BC) = (AB)C (**associative**)
- If A is NxN, I an NxN identity matrix
  - AI = IA = A (I is a **multiplicative identity**)
  - $A^{-1}A = AA^{-1} = I$ if A^{-1} exists

Kronecker Product

- Assume A is MxN, B is PxQ
- C = A⊗B is (M*P)x(N*Q)
  - Replace each A[s,t] by A[s,t]B (replace scalar by matrix)
  - C[i,j] = A[s,t]*B[u,v], i = (s-1)P + u; j = (t-1)Q + v
  - $A^{\otimes k}$ = A⊗A⊗A …. A

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix},$$

Linear Algebra Operations

- Solve for x in Ax = b
  - Gaussian Elimination
  - LU matrix decomposition
- **Inverse** A^{-1} of A where $AA^{-1} = A^{-1}A = I$
- **Determinant** of A, |A|
  - Cramer's rule for 2x2: A[1,1]A[2,2]-A[1,2]A[2,1]
  - Recursively apply for bigger matrices
- **Eigenvectors** and values: Ax = λx

Row Echelon Form

- Matrix A is in **row echelon form** if For row i,
  - A[i,1] = A[i,2] = … A[i,i-1] = 0
  - A[i,i] = 1
  - Rows with all 0's at bottom
- **Reduced row echelon** if A[i,i] only non-zero
- Any matrix can be converted to row echelon:
  - For i=1 to N
    - Find first row j (j ≥ i) with A[j,i] != 0
    - Swap rows i and j
    - Divide each element of new row i by A[i,i]
    - For k>i, multiply row i by A[k,i] and subtract from row k
- Basis for Gauss Seidel method to solve Ax=b

Subspaces and Rank

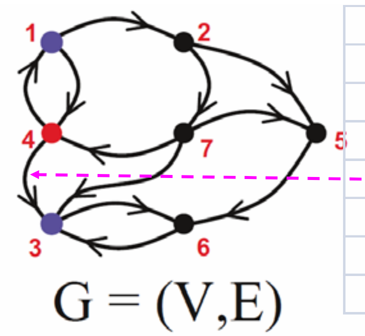
- A set of vectors is **linearly independent** if no one is a weighted sum of the others
- **Rank** of a matrix = # of independent rows (or columns)
  - Compute by forming row echelon & count non-zero rows
- **Full rank**: when all rows independent
- NxN matrix invertible iff rank = N

Infinite Matrix Sums

- Common problem: compute $D = \sum_{k=0,\infty} A^k$
  - $D = I + A + A^2 + A^3 + ...$
- Instead look at $D - DA =$
  - $I + A + A^2 + A^3 + ...\ \ - A - A^2 - A^3 - .... = I$
- Thus $I = D - DA = D(I - A)$
- Or $D = (I - A)^{-1}$
- Often nasty to compute accurately
  - Does not converge
- Alternative if all we care about is whether D[i,j] is 0 or not
  - To compute $D = I + \alpha A + (\alpha A)^2 + (\alpha A)^3 + ...$
  - Compute $D = (I - \alpha A)^{-1}$

Graphs as Matrices

Adjacency Matrix

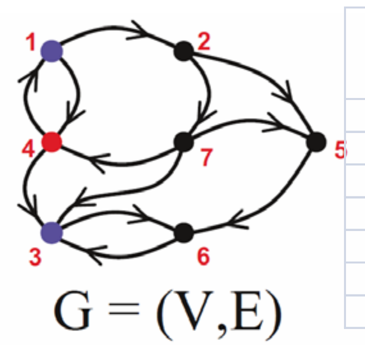


A	1	2	3	4	5	6	7
1	**0**	**1**	**0**	**1**	**0**	**0**	**0**
2	**0**	**0**	**0**	**0**	**1**	**0**	**1**
3	**0**	**0**	**0**	**0**	**0**	**1**	**0**
4	**1**	**0**	**1**	**0**	**0**	**0**	**0**
5	**0**	**0**	**0**	**0**	**0**	**1**	**0**
6	**0**	**0**	**1**	**0**	**0**	**0**	**0**
7	**0**	**0**	**1**	**1**	**1**	**0**	**0**

Book Chap. 7.9

A[u, v] = 1 is edge *from* vertex u *to* vertex 4

Simple Adjacency Properties

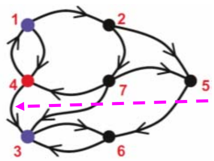


A	1	2	3	4	5	6	7	out-deg	in-deg	deg
1	**0**	**1**	**0**	**1**	**0**	**0**	**0**	2	1	3
2	**0**	**0**	**0**	**0**	**1**	**0**	**1**	2	1	3
3	**0**	**0**	**0**	**0**	**0**	**1**	**0**	1	3	4
4	**1**	**0**	**1**	**0**	**0**	**0**	**0**	2	2	4
5	**0**	**0**	**0**	**0**	**0**	**1**	**0**	1	2	3
6	**0**	**0**	**1**	**0**	**0**	**0**	**0**	1	2	3
7	**0**	**0**	**1**	**1**	**1**	**0**	**0**	3	1	4

Out-degree(u) = sum *across* row u
In-degree(v) = sum *down* column v

Adjacency Observations

- Undirected graph: A[u, v] = A[v, u]
  - Matrix is symmetric
- Weighted graph: A[u, v] = weight on (u,v)
- Transpose: $A^T[v,u] = A[u,v]$
  - != 0 if edge from u to v



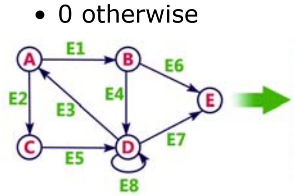
A	1	2	3	4	5	6	7
1	**0**	**1**	**0**	**1**	**0**	**0**	**0**
2	**0**	**0**	**0**	**0**	**1**	**0**	**1**
3	**0**	**0**	**0**	**0**	**0**	**1**	**0**
4	**1**	**0**	**1**	**0**	**0**	**0**	**0**
5	**0**	**0**	**0**	**0**	**0**	**1**	**0**
6	**0**	**0**	**1**	**0**	**0**	**0**	**0**
7	**0**	**0**	**1**	**1**	**1**	**0**	**0**

AT	1	2	3	4	5	6	7
1	0	0	0	1	0	0	0
2	1	0	0	0	0	0	0
3	0	0	0	1	0	1	1
4	1	0	0	0	0	0	1
5	0	1	0	0	0	0	1
6	0	0	1	0	1	0	0
7	0	1	0	0	0	0	0

Incidence Matrices

- Assume G has V vertices and E edges
- **Incidence matrix** M(G) is VxE where
  - Assuming edge e is (u, w), and v a vertex,
  - If M is undirected: M[v,e]=
    - 1 if e=(v,x) or (x,v) (e is "incident on" v)
    - 0 otherwise
  - If M is directed: M[v,e] =
    - 1 if v = u (edge e starts from v)
    - -1 if v = w (edge e ends at w)
    - 0 otherwise

Book Chap. 7.1,7.2



	E1	E2	E3	E4	E5	E6	E7	E8
A	1	1	-1	0	0	0	0	0
B	-1	0	0	1	0	1	0	0
C	0	-1	0	0	1	0	0	0
D	0	0	1	-1	-1	0	1	1
E	0	0	0	0	0	-1	-1	0

http://btechsmartclass.com/DS/images/Graph%20Incidence%20Matrix.jpg

Incidence Matrix Properties

- Every column has exactly two 1s
- Sum over a row = degree of that vertex
- Row with all 0's is an isolated matrix
- Parallel edges (same source and destination) have identical columns
- Permuting any two rows or columns simply "relabels" vertices or edges
- If graph has two disconnected subgraphs, it can be reordered into block-diagonal form

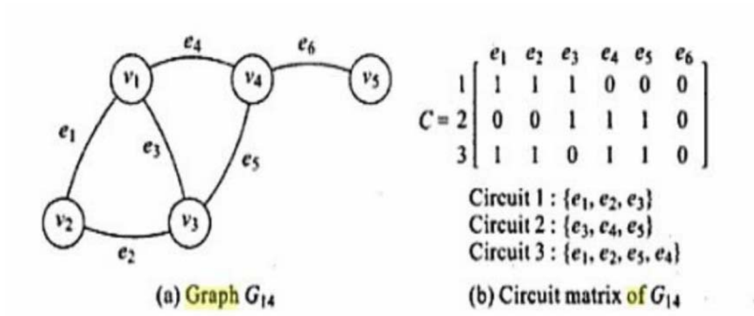
M1	**0**
0	**M2**

- Two graphs are **isomorphic** if their incidence matrices differ only by row/col permutation
- If G is connected, Incidence matrix has rank N-1
- An (n-1)x(N-1) submatrix of M(G) is non-singular iff the N-1 columns of subgraph form a spanning tree

Circuit Matrix

- **Circuit Matrix** B(G) = QxE;
  - Q = # circuits (closed walk where each vertex appears once)
  - B[c,e] = 1 if circuit c includes edge e

Book Chap. 7.3-7.5



https://image.slidesharecdn.com/graphrepresentation-120903115144-phpapp01/95/graph-representation-16-728.jpg?cb=1346673176

Circuit Matrix Properties

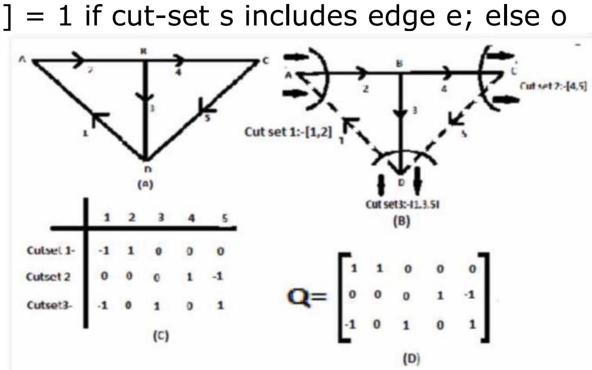
- Each row defines a circuit
- Number of 1s in a row is length of circuit
- Permuting any two rows or columns simply "relabels" circuits or edges
- Column of all 0s corresponds to an edge that's not part of a circuit
- A row can have one 1 – for a self-loop
- If G has two disconnected subgraphs, it can be reordered into block-diagonal form

B1	**0**
0	**B2**

- If B is circuit matrix of incidence matrix A, then $AB^T = B^TA = 0 \pmod 2$
- If G connected then C(G) has rank E-N+1

Cut-Set Matrix

- **Cut-Set Matrix** C(G) = SxE; **Book Chap. 7.6**
  - S = # cut-sets (where a cut-set is a set of edges whose removal breaks graph in two)
  - S[s,e] = 1 if cut-set s includes edge e; else o



https://enknowledges.blogspot.com/2014/10/what-is-cut-set-matrix-in-btech.html

Cut-Set Matrix Properties

- Each row defines a cut-set of edges
- Permuting any two rows or columns simply "relabels" cut-sets or edges
- Column of all 0s corresponds to an edge that's not part of a circuit
- Parallel edges (same source and destination) have identical columns
- Rank of C(G) = rank of A(G)
- If C is cut-set matrix of circuit matrix B, then $CB^T = B^TC = 0 \pmod 2$

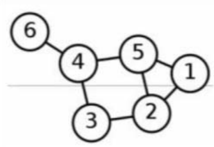
Path Matrix

Book Chap. 7.8

- **Path Matrix** by convention: P(G) is NxN
  - P(u,v) = 1 if a path from u to v; 0 otherwise
- **Path Matrix** from book: $P(G)_{(u,v)}$ = HxE;
  - Separate matrix for each pair of vertices
  - H = # of paths in G between vertices u and v
  - P[p,e] = 1 if path p includes edge e, and 0 otherwise

Laplacian Matrix of a Graph

- **Laplacian** L of simple graph G is D – A
  - G simple: undirected, no self loops,
  - D = degree matrix D[i,i] = degree of vertex i
  - A = Adjacency matrix
  - L[i,j] (a symmetric matrix)
    - = degree(v[i]) if i=j
    - = -1 if i != j and i adjacent to j

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2&0&0&0&0&0\\ 0&3&0&0&0&0\\ 0&0&2&0&0&0\\ 0&0&0&3&0&0\\ 0&0&0&0&3&0\\ 0&0&0&0&0&1 \end{pmatrix}$	$\begin{pmatrix} 0&1&0&0&1&0\\ 1&0&1&0&1&0\\ 0&1&0&1&0&0\\ 0&0&1&0&1&1\\ 1&1&0&1&0&0\\ 0&0&0&1&0&0 \end{pmatrix}$	$\begin{pmatrix} 2&-1&0&0&-1&0\\ -1&3&-1&0&-1&0\\ 0&-1&2&-1&0&0\\ 0&0&-1&3&-1&-1\\ -1&-1&0&-1&3&0\\ 0&0&0&-1&0&1 \end{pmatrix}$

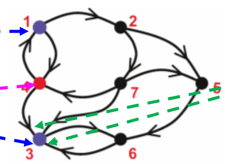
https://en.wikipedia.org/wiki/Laplacian_matrix

Changing the Matrix Operations

Matrix Vector Product

- Conventional y=Mx, M is NxN, |x|,|y|=N
  - $y[i] = \Sigma_{j=1,\ldots N} M[i,j]*x[j]$
  - $= M[i,1]*x[1] + M[i,2]*x[2] + \ldots M[i,N]*x[N]$
- Consider $y = A^T x$,
  - A adjacency matrix
  - x a bit vector of vertices
  - y[i] = # of edges ***into*** vertex i from any vertex in x
  - if y[i]>0, then vertex i ***reachable*** from any vertex in x

y	AT	1	2	3	4	5	6	7	x
1	1	0	0	0	1	0	0	0	0
0	2	1	0	0	0	0	0	0	0
1	3	0	0	0	1	0	1	1	0
0	4	1	0	0	0	0	0	1	1
0	5	0	1	0	0	0	0	1	0
0	6	0	0	1	0	1	0	0	0
0	7	0	1	0	0	0	0	0	0



y	AT	1	2	3	4	5	6	7	x
1	1	0	0	0	1	0	0	0	0
0	2	1	0	0	0	0	0	0	0
2	3	0	0	0	1	0	1	1	0
0	4	1	0	0	0	0	0	1	1
0	5	0	1	0	0	0	0	1	0
0	6	0	0	1	0	1	0	0	1
0	7	0	1	0	0	0	0	0	0

There are 2 edges into vertex 3
if we start at 4 or 6, and 1 edge to vertex 1

Changing the Operators

- Conventional y=Mx, M is NxN, |x|,|y|=N
  - $y[i] = \Sigma_{j=1,\ldots N} M[i,j]*x[j]$
  - $= M[i,1]*x[1] + M[i,2]*x[2] + \ldots M[i,N]*x[N]$
- But what if "*" is "AND", and "+" is "OR"
  - Then y[i]=1 if any AND(M[i,j], x[j]) is 1
  - Or y[i]=1 if there is some j M[i,j]=x[j])=1
- Consider $y = A^T x$ where *=AND, +=OR
  - x a bit vector of vertices
  - y is now **bit vector** of vertices *reachable* from x
- But this (almost) one step of BFS!

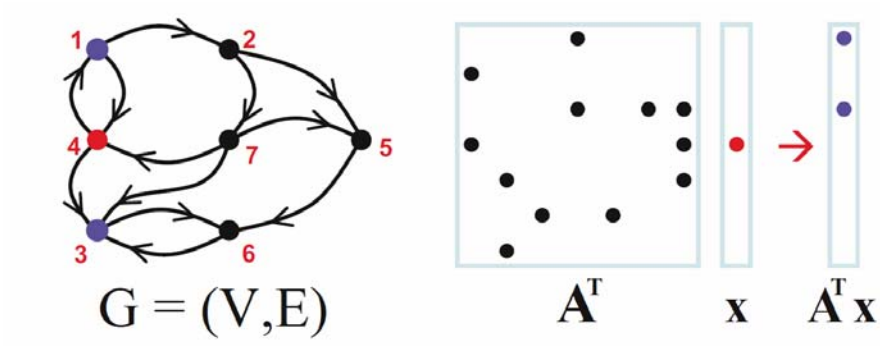


Figure 1.1. Matrix graph duality.
Adjacency matrix **A** is dual with the corresponding graph. In addition, vector matrix multiply is dual with breadth-first search.

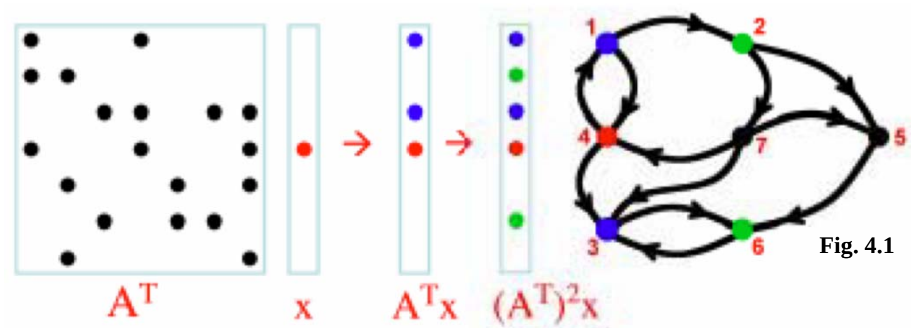
Changing the Operators: Semirings

Rules of linear algebra hold whenever "*" and "+" are functions that form a semi-ring:

- + is **commutative**: a + b = b + a
- Both are **associative**:
  - a + (b + c) = (a + b) + c
  - a * (b * c) = (a * b) * c
- * **distributes** over +:
  - (a + b)*c = a*c + b*c
  - a*(b + c) = a*b + a*c
- Both are **moniods**, i.e. both have identities:
  - a + 0 = a ("0" is additive identity)
  - a*1 = a ("1" is multiplicative identity)
- Additive identity is multiplicative **annihilator**
  - 0*a = a*0 = 0
- Neither + nor * need have inverses

Lets call:
+ the reduction operator
*** the combination operator**

BFS



- Domain: booleans
- + = OR
- * = AND

Minimum Paths

- Assume G has weighted edges (positive only)
- A is adjacency matrix but with ∞ for no edge
- $C_k[u,v]$ = min distance from u to v in *exactly* k steps
  - $C_1[u,v] = A$
- Now assume mat mult ◇ + = min, * = +
- $(A^{\diamond}A)[u,v] = \min_{w=1,N}(A[u,w]+A[w,v]) = A^{\diamond 2}$

  min distance from u to v thru w
- Thus $C_2 = A^{\diamond 2}$; $C_3 = A^{\diamond 3}$; …
- $\min_{i=0,\infty} C_i\,[u,v]$ = min distance from u to v

Useful Semi Rings

+: Reduction Operation			+: Reduction Operation			Sample Usage
Function	Domain	Identity	Function	Domain	Identity	
Normal Add	Ints, floats	0	Normal Multiply	Ints, floats	1	Linear Algebra
OR	Boolean	0	AND	Boolean	0	BFS
min	Ints, floats	∞	Normal Add	Ints, floats	0	Minimum paths