STAT 534 Lecture 4.1

Shortest path algorithms and graph separation

mmp@stat.washington.edu

1 Shortest path algorithm

SHORTEST-PATH(A)

Input graph $\mathcal{G} = (V, \mathcal{E})$, node $A \in V$.

Data structures Each node $U \in V$ has a label l which will store the length of the shortest path between A and U.
The algorithm also uses the *first in/first out priority queue* Q

Initialize $Q = \{A\}$, $l(A) = 0$, $l(U) = \infty$ for $U \neq A$

While Q not empty

1. dequeue U from Q
2. for $W \in n(U)$ *// explore neighbors of* U
   if $l(W) == \infty$ *// found unvisited node*
   (a) $l(W) = l(U) + 1$
   (b) enqueue W

Output $l(U)$ for all $U \in V$

2 Set separation algorithm

A small modification of the above algorithm can verify that two sets are disconnected.

It is assumed that $\mathcal{G}$ below is obtained from the moralized ancestral graph for some independence relationship $X \perp Y|Z$ in an original graph $\mathcal{G}_{orig}$ by removing the nodes in Z.

If in this graph there is no connected component that contains nodes from both X and Y, then $X \perp Y|Z$ holds in the original graph. This is what the next algorithm will check.

SETSEPARATION(X, Y)

Input graph $\mathcal{G} = (V, \mathcal{E})$, disjoint sets $X, Y \subseteq V$.

Data structures Each node $U \in V$ has a label l which will store some auxiliary data. The algorithm also uses the *first in/first out priority queue* Q

Initialize $l(U) = \infty$ for all U, $Q = \emptyset$

While there is a node $A \in X$ with $l(A) = \infty$

// find all nodes reachable from A

1. $Q = \{A\},\ l(A) = 0$

While Q not empty

(a) dequeue U from Q

(b) for $W \in n(U)$ *// explore neighbors of U*
if $l(W) == \infty$ *// found unvisited node*

i. if $W \in Y$ **Output** $\not\perp$ and **Stop**

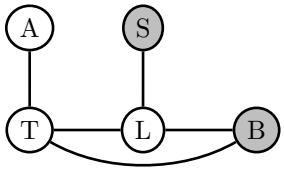
// else

ii. $l(W) = l(U) + 1$

iii. enqueue W

Output $\perp$ and **Stop**

Example Continuation of the chest clinic example. We will test that S not separated from B in the graph



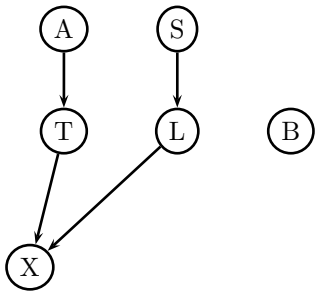
$X = \{S\}$, $Y = \{B\}$, $l = \infty$ for all $U \in V$

1. $l(S) = 0$, enqueue S, $Q = (S)$
2. dequeue S; $n(S) = \{L\}\, l(L) \leftarrow 1$, enqueue L, $Q = (L)$
3. dequeue L; $n(L) = \{B, T\}$

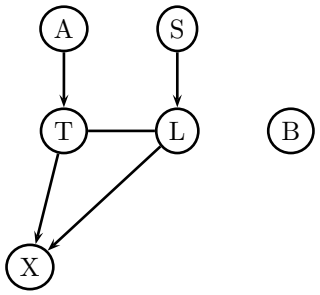
(a) $B : B \in Y$, **Stop:** Original independence invalid

Example Show D-separation $AS \perp B \,|\, X$ in the Chest clinic example

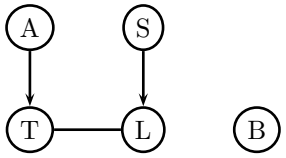
1. Ancestral set A, T, L, S, X, B, ancestral graph



2. V-structures in ancestral graph X; moral graph



3. Delete conditioning node X



4. SetSeparation(AS, B)

   (a) $l(A) \leftarrow 0$, enqueue A, $Q = (A)$
   (b) dequeue A, $n(A) = \{T\}$, $l(T) \leftarrow 1$, enqueue T, $Q = (T)$
   (c) dequeue T, $n(T) = \{A, L\}$,A already visited, $l(L) \leftarrow 2$, enqueue L, $Q = (L)$
   (d) dequeue L, $n(L) = \{T, S\}$, T already visited, $l(S) \leftarrow 3$, enqueue S, $Q = (S)$
   (e) dequeue S, $n(S) = \{L\}$, L already visited, $Q = ()$
   (f) no more unvisited nodes in X, **Stop** Original independence valid

3 Reading

Shortest Path Algorithms

Lecture 4.1. Additionally, "Single source shortest path" in CRLS and West chapters 2.1 and 2.3.

Further reading: "All shortest paths algorithms", "Dynamic programming" from CRLS, and the highly popular *Contraction hierarchies* shortest path routing algorithm (posted)

Minimum Spanning Tree Algorithms

CRLS "Minimum Spanning Tree Algorithms" chapter (handed out in class), KF Appendix, CRLS "Data structures for disjoint sets" chapter (handed out in class), West chapters 2.1 and 2.3.

Further reading: Randomized exact MST (e.g. Karger, Klein & Tarjan), Randomized Approximate MST, Distributed MST (e.g. GHS algorithm) *(references to be added)*

Spanning Trees – other topics

The most interesting source for this is West - Chapter 2: 2.2 contains the Matrix Tree Theorem and other topics related to enumeration on trees.

Further reading from West: Chordal graphs and perfect graphs 8.1, Matroids 8.2, Random graphs 8.5 (the "uniform sampling over edges" Erdos-Renyi model), Eigenvalues of graphs (you will encounter the Laplacian again) 8.6

Matrix tree theorem next section in these notes and "Tractable Bayesian learning of tree distributions" (posted)

Chordal graphs

Lecture 6, KF chapter 10 (and 4.5)

Further reading: "How chordal graphs work" by Terry McKee, handed out in class

4 The Matrix Tree Theorem

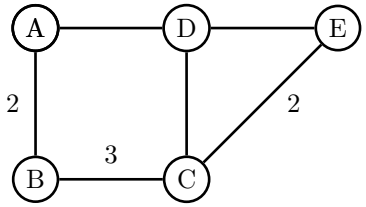
Theorem 1 (Matrix Tree Theorem) *Let $\mathcal{G} = (V, \mathcal{E})$ be a loopless (multi)graph, and denote by $a_{uv} \in \{1, 2, \ldots\}$ the number of edges between nodes u and v in V.*

Let $A = [a_{uv}]_{u,v\in V}$, $d_v = \sum_{u\in V} a_{uv}$, $L = \mathrm{diag}\{d\} - A$ (these are called respectively the adjacency matrix*,* vector of degrees *and* Laplacian *of graph $\mathcal{G}$).*

Then, the number of distinct spanning trees of $\mathcal{G}$ is given by $\tau\mathcal{G} = (-1)^{i+j} \det L_{-i,-j}$, where $L_{-i,-j}$ denotes the matrix L above with row i and column j deleted.

Remarks 1. The determinant of L itself is 0. The theorem implies that all minors of $\det L$, e.g. all $\det L_{-i,-j}$ have the same value, up to a ± 1 factor.

2. About the number of trees in a *multigraph.*
Take for example the graph



where some edges have been labeled. The label on the edges represents (here) the number of edges between the two nodes, if it's larger than 1 (for instance, there are 2 edges between A and B).

Now consider the spanning tree given by $\mathcal{E}_T = \{AB, BC, CD, DE\}$. Since the edge AB has multiplicity 2, and BC has multiplicity 3, it means that there are $2 \times 3 = 6$ distinct spanning trees over the set $\mathcal{E}_T$, and we can think of this as the *multiplicity of* $\mathcal{E}_T$, or simply the multiplicity of tree T.

Hence the number of distinct spanning trees in a multigraph, is the of multiplicities of all the spanning trees T of the corresponding *simple* graph.

$$\tau(\mathcal{G}) \overset{def}{=} \sum_{T \mathrm{sp.tree}} \prod_{uv\in\mathcal{E}_T} a_{uv}$$

3. For a disconnected graph, the number of *spanning trees* is 0 and this is what the MTT will count. [Exercise] Generalize the MTT to disconnected graphs to count the spanning forests.

4.1 Distribution over trees

[This is covered in detail in [Meila&Jaakkola]
The MTT does not depend on the fact that the edge multiplicities are integer, and can immediately be generalized to weighted graphs.

$$\tau(W) \;=\; \sum_{T \text{sp.tree}} \prod_{uv \in \mathcal{E}_T} w_{uw} \;=\; (-1)^{i+j} \det L_{-i,-j}(W) \tag{1}$$

where $W = [w_{uv}]$, $w_{uw} \geq 0$ and $L(W)$ is defined as above, with W instead of A.

What does $\tau(W)$ represent now?

One interpretation is probabilistic. We can define a *factored* distribution P over the trees of $\mathcal{G}$ by

$$P(T) \;=\; \begin{cases} \frac{1}{\tau(W)} \prod_{uv \in \mathcal{E}_T} w_{uw} & \text{if} T \text{spanning tree} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This distribution is reminiscent of a MRF, but has the remarkable property that the normalization constant is $\tau(W)$, always tractable, no matter what the structure of the underying $\mathcal{G}$. Just like in a MRF, the "potential" w_{uv} does not represent the marginal probabiliity of the respective edge uv of belonging to the tree.

Re-weighting edges Let now $F = [f_{uv}]$ be a non-negative function over the edges, different from W. Define

$$F(T) \;=\; \prod_{uv \in \mathcal{E}_T} f_{uv} \tag{3}$$

For example, $F(T) = e^{-energy(T)}$ where the energy comes from another process than the weights. Or, F represents the likelihood of the edges/trees, while W represents their prior.

In any case, it is straightforward (EXERCISE) to compute the expectation:

$$E_P[F(T)] \;=\; \sum_T P(T)F(T) \;=\; \frac{\tau([w_{uv} f_{uv}]_{uv})}{\tau(W)} \tag{4}$$

Expectation over additive weights Let again $F = [f_{uv}]$ be a non-negative function over the edges, different from W, but define

$$F(T) \;=\; \sum_{uv \in \mathcal{E}_T} f_{uv} \tag{5}$$

This is the case where F represents the *cost* of edges/trees, while W represents their probability.

Surprisingly, this expectation can also be computed easily

$$E_P[F(T)] \;=\; \sum_T P(T)F(T) \;=\; \sum_T \left(\sum_{uv \in \mathcal{E}_T} f_{uw} \right) \prod_{uv \in \mathcal{E}_T} w_{uw}. \tag{6}$$

To see that, define new weights $\tilde{w}_{uv} = w_{uv} e^{t f_{uv}}$, with $t \geq 0$. Then, it is easy to check that

$$\frac{1}{\tau(W)} \left. \frac{d\tau(\tilde{W})}{dt} \right|_{t=0} \;=\; E_P[F(T)] \tag{7}$$

using the fact that $d\tilde{w}_{uv}/dt = f_{uv}\tilde{w}_{uv}$. It remains to compute the derivative of $\tau(\tilde{W})$ w.r.t. t. This can be done by taking into account that, for any square matrix $A = [a_{ij}]$,

$$\frac{\partial \det A}{\partial a_{ij}} \;=\; A^*_{ij} \tag{8}$$

where A^*_{ij} represents the i,j-*minor* of A.

Assuming that τ is computed by removing the last row and column of L we have (details skipped), and defining $Q = L_{-n,-n}$ and M as

$$\begin{array}{rcl} M_{uv} & = & (Q^{-1})_{uu} + (Q^{-1})_{vv} - 2(Q^{-1})_{uv},\ u, v < n \\ M_{nv} & = & M_{vn} \;=\; (Q^{-1})_{vv},\ v < n \\ M_{vv} & = & 0 \end{array} \tag{9}$$

we obtain the expectation as

$$E_P[F(T)] \;=\; \sum_{u<v} f_{uv} w_{uv} M_{uv} \;=\; \mathrm{trace}[L_{-n,-n}([f_{uv}w_{uv}]) L^{-1}_{-n,-n}(W)] \tag{10}$$