Lecture 11 — Weighted Graphs, Spanning Trees and Shortest Paths

Weighted Graphs

Definition 1. A weighted graph $G = (\mathcal{N}, \mathcal{A})$ is one where each arc $(i, j) \in \mathcal{A}$ has associated with it a weight w_{ij}.

The concept of a weighted graph is extremely useful. The weights can be thought of, for example, as the cost of sending a message down a particular arc. (Not necessarily a monetary cost but some combination of time and distance for example). Weighted graphs can be used to formulate the shortest path problem for routing packets. Let us first look at a different problem, that of the minimum weight spanning tree (MST).

Minimum Weight Spanning Tree

Recall from last lecture that a spanning tree of a connected graph $G = (\mathcal{N}, \mathcal{A})$ is a subgraph $G' = (\mathcal{N}', \mathcal{A}')$ which is a tree and in which all the nodes in $\mathcal{N}$ are also in $\mathcal{N}'$.

Definition 2. A *minimum weight spanning tree* (MST) of a weighted graph $G = (\mathcal{N}, \mathcal{A})$ with arc weights w_{ij} is the spanning tree $G' = (\mathcal{N}', \mathcal{A}')$ which minimises the sum $\sum_{(i,j)\in\mathcal{A}'} w_{ij}$.

Definition 3. A *fragment* of a graph G is a subtree (a subgraph which is a tree) of an MST of G.

Proposition 1. *Given a fragment $F = (\mathcal{N}', \mathcal{A}')$ of a graph $G = (\mathcal{N}, \mathcal{A})$, let $\alpha = (i, j)$ be the minimum weight arc which has the property $i \in \mathcal{N}'$ and $j \in \mathcal{N} - \mathcal{N}'$. F extended by adding the arc (i, j) and the node j is also a fragment.*

This proposition, if true, immediately suggests an algorithm for creating an MST for a graph G by expanding the fragment F until it becomes an MST.

Proof. M is the MST of which F is a subtree. Choose some $\alpha = (i, j)$ as described in proposition 1. If $\alpha = (i, j) \notin M$ (where α is the minimum weight arc as described above) then there must be a cycle formed by the arcs of M and α. Since $j \notin F$ by definition, then there must exist some arc $\beta = (k, l) \in M$ where $\beta \neq \alpha$ which is part of the cycle and where $k \in F$ and $l \notin F$. Deleting β from M and adding α instead must therefore result in a spanning tree M' (since M' is a subgraph of G which has no cycles and has the same number of nodes and arcs as M – from last lecture's lemma). Since α has lower weight than β then M' has a lower total weight than M which is a contradiction. Therefore, our original assumption that $\alpha = (i, j) \notin M$ cannot be true and the proposition is proved. □

Two algorithms use this proposition to build MSTs from a graph. The Prim-Dijkstra algorithm starts with an arbitrary node as the initial fragment and adds nodes and arcs as described by the above proposition until the fragment becomes an MST.

Kruskal's algorithm starts with each node as a fragment and, iteratively, choses the arc which is the minimum weight arc which does not form a cycle. Ideally, we would be able to use Kruskal's algorithm in a distributed manner and have each fragment add arcs until all the fragments are joined. Kruskal's algorithm requires a second proposition before we can use it in a distributed manner. An MST is not necessarily unique. Consider, for example, a three node graph with each arc weight as 1 as seen in figure 2

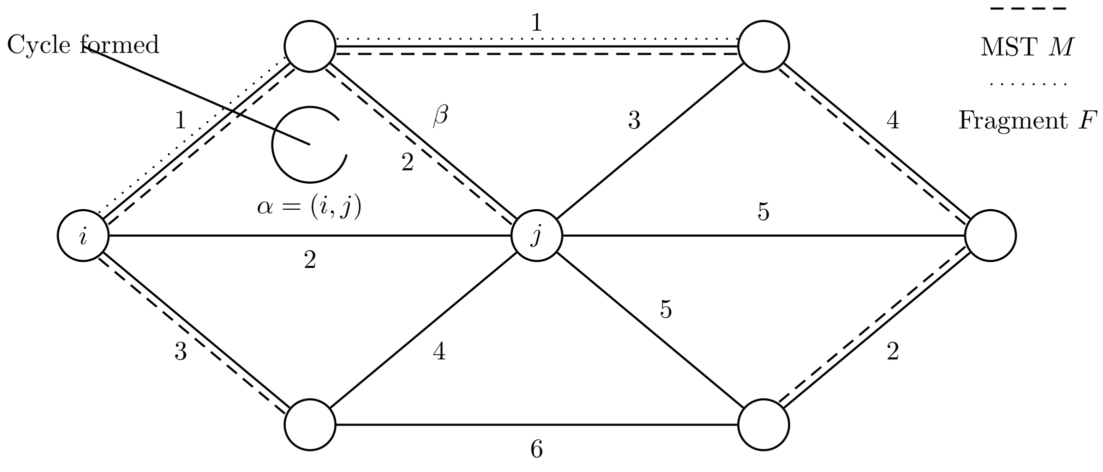

Figure 1: MST: (Example taken from Bertsekas and Gallagher)

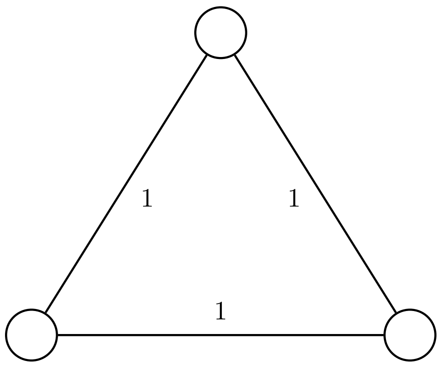

Figure 2: Graph without unique MST. (Taken from B & G)

Proposition 2. *If all arc weights in a graph are distinct then there exists a unique MST.*

Proof. Suppose that two distinct MSTs exist M and $M\prime$. Let α be the smallest arc which belongs to one but not both. Assume wlog that α belongs to M. The situation is shown in figure 3. Consider the graph formed by the union of M' and α. Clearly the graph must contain a cycle. At least one arc of this cycle must not belong to M (or M would itself contain a cycle). Since α must have a smaller weight than β by definition then, by deleting β from M' and replacing it with α then M' will be a spanning tree of smaller weight than it originally had. This is a contradiction since our original assumption was that both trees were MST. □

If a unique MST exists we can use the distributed version of Kruskal's algorithm. That is, each node is considered as a fragment and each fragment adds simultaneously, the minimum weight arc which is outgoing from that fragment. (Which must be part of an MST from the earlier proposition, and since the MST is unique, all nodes added must be part of the same MST). New arcs will be added until all the fragments are joined.

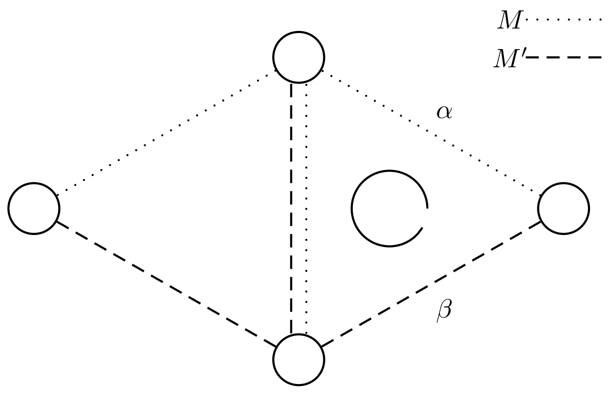

Figure 3: Proof of MST uniqueness. (Taken from B & G)

Directed Graphs and Shortest Paths

Definition 4. A *directed graph* or *digraph* $G = (\mathcal{N}, \mathcal{A})$ is a finite, nonempty set $\mathcal{N}$ of nodes and a collection of *ordered* pairs of distinct nodes from $\mathcal{N}$. (Each member of $\mathcal{A}$ is called a *directed arc.*

The terms *directed walk* and *directed path* have the expected meanings — digraph equivalents of *walk* and *path* where the direction of the arcs is taken into account. A subtle difference should be observed when considering a directed cycle.

Definition 5. A *directed cycle* is a directed walk $(n_1, \dots, n_l)$ with $n_1 = n_l$ where $l > 2$ and no repeated nodes other than 1 and l. Note that this is slightly different to the notion of a cycle in an undirected graph.

Note the subtle difference between this and the definition of a cycle in a non-directed graph. (Note also that since we have disallowed arcs which start and end at the same node, if $(n_1, \dots, n_l)$ with $n_1 = n_l$ where $l > 2$ and there are repeated nodes other then 1 and l then the directed walk must be made of several cycles.

We can now define the problem of finding a shortest path from O to D as finding the directed path $(n_1, \dots n_l)$ where $n_1 = O$ and $n_l = D$ which minimises the sum $\sum_{i=1}^{l-1} w_{n_i n_{i+1}}$ where w_{ij} is the weight of the arc from node i to node j. Two methods of solving this problem will be discussed in the next lecutue: Dijkstra's algorithm and the Bellman-Ford algorithm. Both of these are in common use, both in mathematics in general, and in route-finding on the internet in specific.

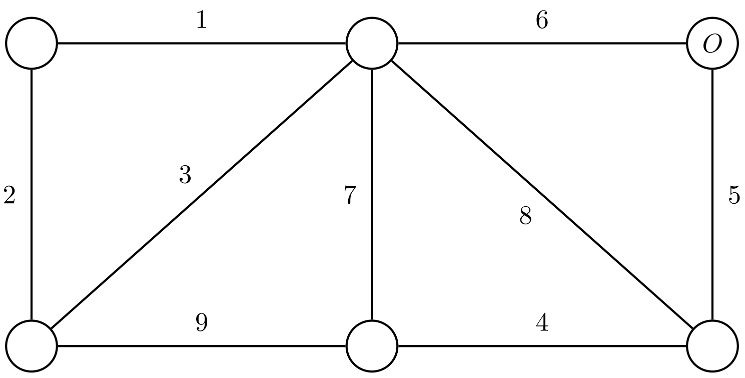

Figure 4: Figure for testing Prim-Dijkstra vs Kruskal's Algorithm. (Taken from B & G)