Network Problem Models

Author | Ishank Juneja

May 27, 2018

1 Introduction

Some of the problems modelled as Network problems are :

1. Transportation Problem
2. Assignment Problem
3. Minimum Spanning Problem
4. Shortest path problem
5. Maximum flow problem
6. Min cost flow problem

The commonality as they are all associated with an underlying Graph.

2 Minimum Spanning Tree Problem

To obtained a spanning tree for a given undirected, connected graph such that sum of edge weights is minimum.

Prim's Algorithm

The algorithm maintains two arrays $S_1 = \{\text{Set of nodes which have been added to spanning tree}\}$ and $S_2 = \{\text{Rest of the nodes}\}$.
To start off, we put the arc with minimum weight into the spanning tree.
In every subsequent Iteration the aim is to add nodes from S_2 to S_1. We do so by checking for all possible connection from nodes in S_1 to those in S_2.
From all these possible connections we pick the one with the least edge weight and move the corresponding end point node from S_2 to S_1. Ties in edge weights are broken arbitrarily.

1. $S_1 = \{1, 2\}, S_2 = \{3, 4, 5\}$
2. $S_1 = \{1, 2, 3\}, S_2 = \{4, 5\}$
3. $S_1 = \{1, 2, 3, 5\}, S_2 = \{4\}$
4. $S_1 = \{1, 2, 3, 4, 5\}, S_2 = \{\}$

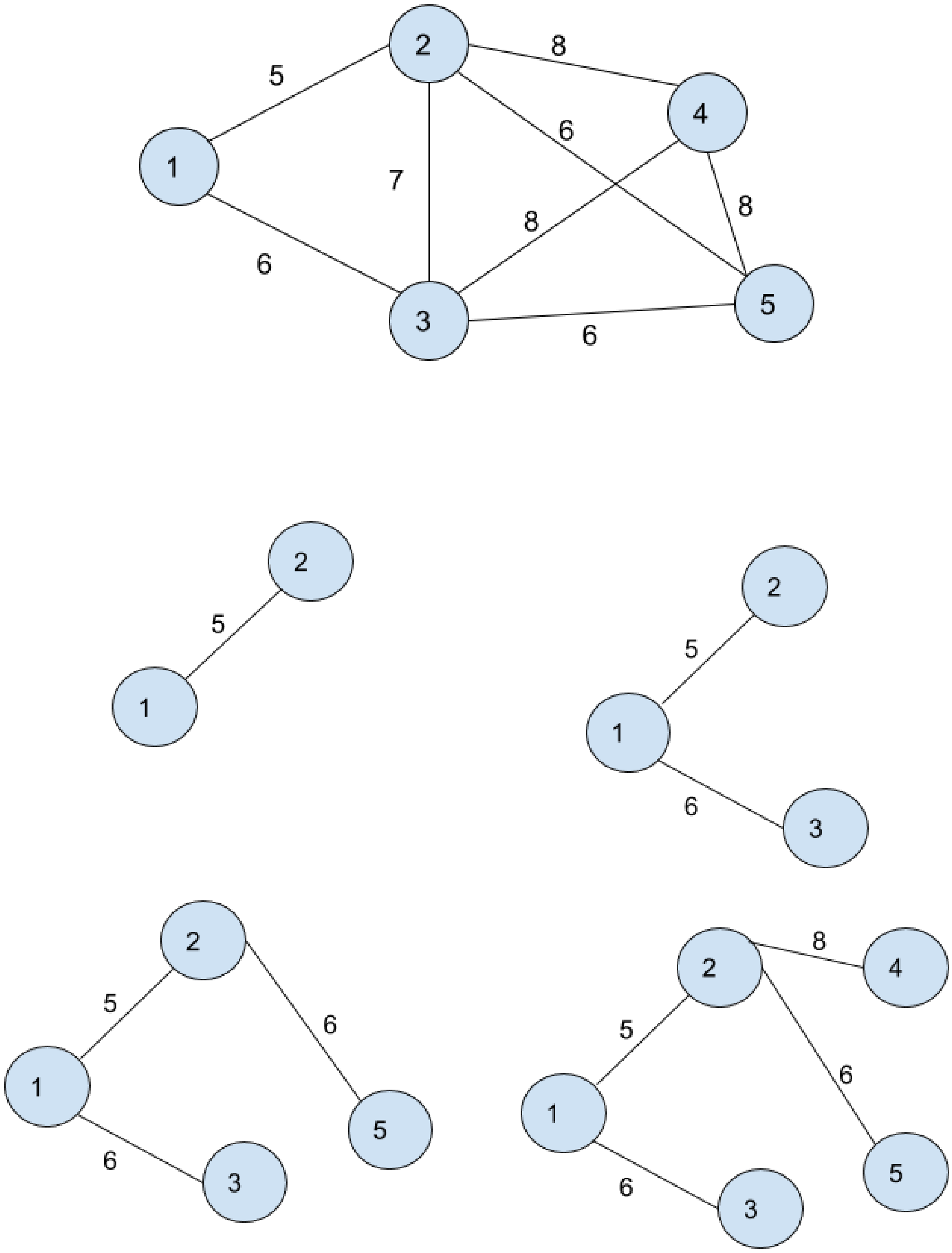

Figure 1: Prim's Algorithm Illustration

Kruskals Algorithm

1. Arrange edge weights in non decreasing order
   1–2 5
   3–5 6
   1–3 6
   2–5 6
   2–3 7
   4–5 8
   2–4 8
   3–4 8

2. Follows roughly the same procedure. Go down the list, any edge encountered is added to the spanning tree if the addition does not create a cycle (Thereby breaking the Tree property).

3. Like Prim's we will get one of the optimal spanning trees.

Reasoning

Cut Optimality Theorem For every edge i–j $\in T^*$, the minimal spanning tree, $c_{ij} \leq c_{kl} \quad \forall \quad k, l$ in the cut obtained by deleting i–j from the tree.
Prim's is a direct implementation of Cut Optimality.

Path Optimality Theorem For every non tree edge (k,l), $c_{ij} \leq c_{kl}, (i, j) \in T^*$.
Kruskals method is a direct implementation of Path Optimality.

3 Shortest path Problem: Dijkstra's Algorithm

Find Shortest path from node 1 to node 7.

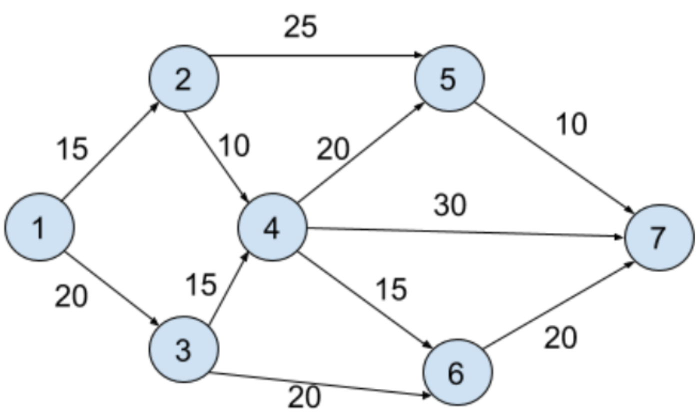

Figure 2: Prim's Algorithm Illustration

1. Start from node 1, cost = 0. Node 1 will now not be visited again. Jump to the next destination with minimum total cost starting from node 1. In this case node 2.

2. At node 2, cost incurred = 15, node 2 will not be visited again, options 3 (d = 20), 4 (d = 25), 5 (d = 40). Hence go to node 3 next.

3. At 3, discover better path for 5, discover 6 and move to 4 next. And so on...

An assumption in the algorithm is that edge weights $c_{ij} \geq 0$

Extensions of Dijkstra's Algorithm

The algorithm solves

- Fixed source to fixed destination (std. case)
- Fixed source to any destination (already solved)
- Any source to given destination (Applying Algorithm in reverse i.e on destination)
- Any source to any destination (N-1 iterations of Algorithm)

Formulation as Programming Problem

We are trying to formulate the original 1 -> 7 shortest path problem as a programming problem.
$X_{ij} = 1$ if arc i–j lies in the shortest path, $= 0$ otherwise.
Minimize $\sum c_{ij} X_{ij}$ subject to the constraints
$X_{12} + X_{13} = 1$. Atleast one of them must be true for there to be a path.
$-X_{12} + X_{24} + X_{25} = 0$ Like a KCL on node 2
$-X_{13} + X_{34} + X_{36} = 0$
$-X_{24} - X_{34} + X_{45} + X_{46} + X_{47} = 0$
$-X_{25} - X_{45} + X_{57} = 0$
$-X_{36} - X_{46} + X_{67} = 0$
$X_{47} - X_{57} - X_{67} = 1$

Uni-modularity Property : Even if we relax Binary assumption solutions will still be (0,1).

3.1 Negative Edge Weights

Solved using Bellman-Ford Algorithm. CS213 course.

4 References

1. Advanced OR NPTEL, G Srinivasan IIT Madras