Algorithm to Find the Minimum Spanning Tree in a Directed Graph

Abhinav Mishra
Indian Institute of
Information Technology
iim2014003@iiita.ac.in

Neelanjana Jaiswal
Indian Institute of
Information Technology
ism2014005@iiita.ac.in

Sannihith Kavala
Indian Institute of
Information Technology
ihm2014004@iiita.ac.in

***Abstract*—A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. There can be many spanning trees possible in a given graph. Minimum spanning tree is the spanning tree where the cost is minimum among all the spanning trees. There also may be a possibility that the minimum spanning tree is not unique. Minimum spanning tree has direct application in the design of networks. It is used in algorithms approximating the travelling salesman problem, multi-terminal minimum cut problem and minimum-cost weighted perfect matching. The algorithms such as Prims and Krushkal are used to find Minimum Spanning Tree but are employable only in class of undirected graphs. We will be analyzing algorithm for finding Minimum Spanning Tree in the case of directed graphs.**

***Index Terms*—Directed Graph, Spanning tree, MST.**

I. Introduction

A minimum spanning tree (MST) or minimum weight spanning tree is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight. That is, it is a spanning tree whose sum of edge weights is as small as possible. For a directed graph the term analogous to spanning tree is spanning arborescence.In graph theory, an arborescence is a directed graph in which, for a vertex u called the root and any other vertex v, there is exactly one directed path from u to v. An arborescence is thus the directed-graph form of a rooted tree, understood here as an undirected graph.

Equivalently, an arborescence is a directed, rooted tree in which all edges point away from the root. Every arborescence is a directed acyclic graph (DAG), but not every DAG is an arborescence. An arborescence can equivalently be defined as a rooted digraph in which the path from the root to any other vertex is unique.

II. Motivation

Finding the minimum spanning tree is a very popular problem in Graph Theory.There are quite a few use cases for minimum spanning trees. One example would be a telecommunications company which is trying to lay out cables in a new neighborhood. If it is constrained to bury the cable only along certain paths (e.g. along roads), then there would be a graph representing which points are connected by those paths. Some of those paths might be more expensive, because they are longer, or require the cable to be buried deeper; these paths would be represented by edges with larger weights.

Minimum Directed Spanning Trees have direct applications in the design of networks, including computer networks, telecommunications networks, transportation networks, water supply networks, and electrical grids. They are invoked as subroutines in algorithms for other problems, including the Christofides algorithm for approximating the traveling salesman problem, approximating the multi-terminal minimum cut problem (which is equivalent in the single-terminal case to the maximum flow problem), and approximating the minimum-cost weighted perfect matching.

III. Algorithm

A. Difference between Directed and Undirected Case

In undirected graphs, every edge is part of some spanning tree. In directed graphs, not all edges are part of DSTs. Edges that are not contained in any DST are said to be useless. The simplest example of useless edges are edges that enter the root r. Other edges may also be useless. Directed versions of the cut and cycle rules that were so instrumental in devising algorithms for finding minimum spanning trees in undirected graphs are no longer valid, even if we assume that all edges are useful. (Finding counterexamples is again left as an exercise.) We thus need to use a new technique to finding MDSTs.

B. Minimum Directed Spanning Trees

Let $G = (V; E; w)$ be a weighted directed graph, where $w : E \rightarrow$ is a cost (or weight) function defined on its edges. Let $r \in V$. A directed spanning tree (DST) of G rooted at r, is a subgraph T of G such that the undirected version of T is a tree and T contains a directed path from r to any other vertex in V . The cost $w(T)$ of a directed spanning tree T is the sum of the costs of its edges, i.e., $w(T) = \sum_{e \in T} w(e)$. A minimum directed spanning tree (MDST) rooted at r is a directed spanning tree rooted at r of minimum cost. A directed graph contains a directed spanning tree rooted at r if and only if all vertices in G are reachable from r. This condition can be easily tested in linear time.

A directed graph is shown in 1 with the edges to be included in minimum cost arborescence is shown in bold. Given a

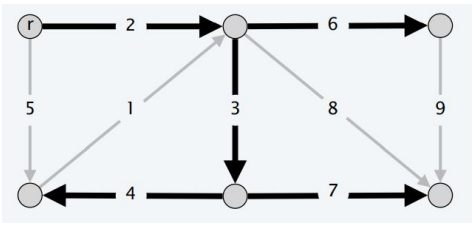


Fig. 1. Bold edges in the above directed graph represents minimum cost arborescence

digraph $G(V, E)$ and with a non-negative cost $c \geq 0$ on each edge e, the problem is to compute an arborescence rooted at r of minimum cost.

C. Properties of MDST

1) Possible multiplicity
   If there are n vertices in the graph, then each spanning tree has $n1$ edges.There may be several minimum spanning trees of the same weight; in particular, if all the edge weights of a given graph are the same, then every spanning tree of that graph is minimum.
2) Uniqueness
   If each edge has a distinct weight then there will be only one, unique minimum spanning tree.

D. Edmonds' Algorithm

In this work we have used Edmonds' algorithm to solve the given problem.

The Edmonds' algorithm takes as input a directed graph $D(V, E)$ where V is the set of nodes and E is the set of directed edges, a distinguished vertex $r \in V$ called the root, and a real-valued weight $w(e)$ for each edge $e \in E$. It returns a spanning arborescence A rooted at r of minimum weight, where the weight of an arborescence is defined to be the sum of its edge weights, $w(A) = \sum_{e \in A} w(e)$.

The algorithm has a recursive description. Let $f(D, r, w)$ denote the function which returns a spanning arborescence rooted at r of minimum weight. We first remove any edge from E whose destination is r. We may also replace any set of parallel edges (edges between the same pair of vertices in the same direction) by a single edge with weight equal to the minimum of the weights of these parallel edges.

Now, for each node v other than the root, find the edge incoming to v of lowest weight (with ties broken arbitrarily). Denote the source of this edge by $\pi(v)$. If the set of edges $P = \{(\pi(v), v) \mid v \in V \setminus \{r\}\}$ does not contain any cycles, then $f(D, r, w) = P$.

Otherwise, P contains at least one cycle. Arbitrarily choose one of these cycles and call it C. We now define a new weighted directed graph $D' = \langle V', E' \rangle$ in which the cycle C is "contracted" into one node as follows:

The nodes of V' are the nodes of V not inC plus a new node denoted v_C.

- If (u,v) is an edge in E with $u \notin C$ and $v \in C$ (an edge coming into the cycle), then include in E' a new edge $e = (u, v_C)$, and define $w'(e) = w(u, v) - w(\pi(v), v)$.
- If (u,v) is an edge in E with $u \in C$ and $v \notin C$ (an edge going away from the cycle), then include in E' a new edge $e = (v_C, v)$, and define $w'(e) = w(u, v)$.
- If (u,v) is an edge in E with $u \notin C$ and $v \notin C$ (an edge unrelated to the cycle), then include in E' a new edge $e = (u, v)$, and define $w'(e) = w(u, v)$. For each edge in E', we remember which edge in E it corresponds to.

Now find a minimum spanning arborescence A' of D' using a call to $f(D', r, w')$. Since A' is a spanning arborescence, each vertex has exactly one incoming edge. Let (u, v_C) be the unique incoming edge to v_C in A'. This edge corresponds to an edge $(u, v) \in E$ with $v \in C$. Remove the edge $(\pi(v), v)$ from C, breaking the cycle. Mark each remaining edge in C. For each edge in A', mark its corresponding edge in E. Now we define $f(D, r, w)$ to be the set of marked edges, which form a minimum spanning arborescence.

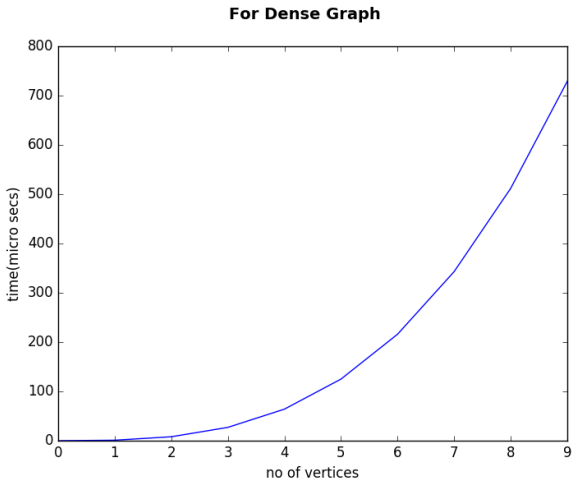
Observe that $f(D, r, w)$ is defined in terms of $f(D', r, w')$, with D' having strictly fewer vertices than D. Finding $f(D, r, w)$ for a single-vertex graph is trivial (it is just D itself), so the recursive algorithm is guaranteed to terminate.

IV. Time Complexity

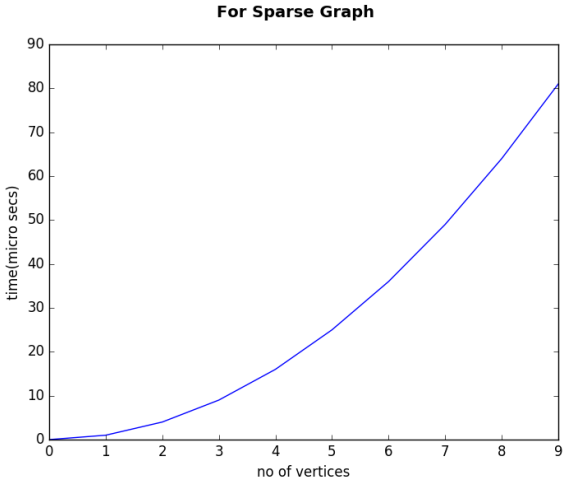
The running time for finding minimum spanning tree using edmond's algorithm is $O(EV)$ where E is the number of edges and V is number of vertices.

V. Graphs

A. For Dense graphs



B. For Sparse graphs



VI. Conclusion

In this paper we implemented Edmond's algorithm for finding Minimum Spanning Tree in Directed Graph with a Complexity of O(EV) where E is number of edges and V is number of vertices.