Project INF442 : Minimum Spanning Trees (MST) *

Juan-Antonio Cordero
juan-antonio.cordero-fuertes@polytechnique.edu

1 Motivation

Spanning trees and Minimum Spanning Trees (MSTs) are fundamental objects in graph theory, and are used in many different applications. In computer science, for instance, MSTs are useful to perform flooding in (wired) computer networks with a minimum number of transmissions. Protocols such as the Spanning Tree Protocol (STP) [12] or the standard IEEE 802.1d build a spanning tree overlay on top of an Ethernet network (layer-2), in order to avoid routing loops. Multicast protocols such as DVMRP (Distance Vector Multicast Routing Protocol) [1] also rely on spanning trees on top of the network graph (layer-3) to deliver multicast flows over a set of destinations.

MST is also used for data science purposes. This project explores the use of MST as a basis for clustering algorithms, and the performance of these algorithms with respect to other heuristics (e.g., k-means).

The project is organized in sections and tasks. These are provided to give you hints on aspects that can be interesting to look at. Tasks 1 and 2 are basic and required for the others; the rest can be explored depending on your interest and will be graded accordingly. You are encouraged to use the provided references and resources as pointers to develop your work.

2 Formal definitions

Given a graph $G = (V, E)$, a **spanning tree** for G, denoted T, is a tree (*i.e.*, a graph in which every pair of vertices is connected by way of one and only one path) such that every vertex $v \in V$ is part of it. In other words, it is a subgraph $T \subset G$ that contains all vertices in G, is connected and is acyclic.

Given a weighted graph $G = (V, E, w)$, where $w(e)$ is the weight of edge $e \in E$, the **minimum spanning tree** is the spanning tree of G with minimum weight.

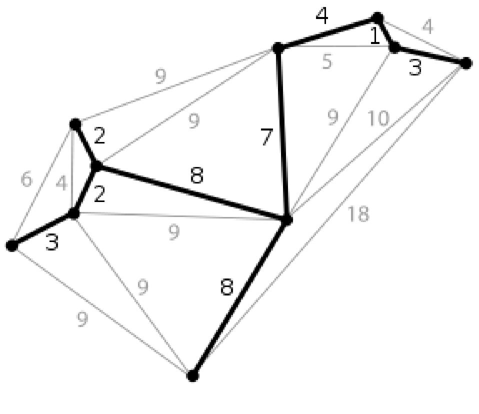

Figure 1: Example of MST over a weighted graph. (Source: Wikipedia)

3 Sequential implementation of classic algorithms

Several algorithms have been proposed to compute, if exists, the MST of a given graph. The main ones are Kruskal [10], Prim [9] and Boruvska [7].

Task 1 Describe and implement the Boruvska and Prim algorithms to compute the MST for a given general graph. Discuss their complexity. Test and discuss their performance with networks of different sizes (n)[1].

Task 2 Describe and implement Kruskal algorithm for a given general graph. As in the previous case, discuss its complexity, test it and evaluate it performance with networks of different sizes (n).

4 Parallel implementation of MST algorithms, and fully distributed computation of MST

For parallel implementations, you can use the Message Passing Interface (MPI) standard and, in particular, the MPI library for C[2].

Prim is the most intuitive approach to MST construction, but it is not easily parallelizable; Boruvska algorithm is less intuitive, but presents better parallel characteristics: most of the research on parallel construction of MSTs relies on this work. Different parallelizations of these algorithms have been proposed in the literature; we propose to explore some of them, e.g. the parallelized versions of Prim and Boruvska described by Kumar [2] and Guang-rong et al. [4], respectively (students may explore other possibilities from the existing literature).

Task 3 Implement parallelized versions of Prim and Boruvska with MPI, and evaluate their communication cost and time performance for different computer networks.

These algorithms (Prim, Boruvska) assume that a central instance has a complete view of the graph and can thus compute the MST. In a real computer network, the MST computation would require a full knowledge of the network topology, the resulting forwarding decisions would need to be implemented afterwards in each of the involved computers. Note that explored parallelized versions of the algorithms rely on the same setup – a central instance able to perform its computation by using several concurrent processors.

In practice, central computation with full knowledge of the graph is not feasible in many computer networks. Instead, distributed algorithms that can be used independently in each computer of the network are preferred. Gallager [6] proposed a distributed computation of the Minimum Spanning Tree (MST) on a network, in which nodes only need to know about their neighbors in order to run the heuristic. Neighbor sensing is typically achieved in computer networks by allowing every node to periodically sending *presence* messages (denominated *Hello* messages) to neighbors reachable through all its interfaces (outgoing edges).

Task 4 Implement Gallager's algorithm using MPI. You can emulate neighbor sensing by enabling the root to distribute neighborhood information to every node in the network. Discuss the complexity of this algorithm and the communication costs that it incurs for a network with n nodes.

5 MST for clustering

MST can be used for data clustering. This requires that data items are connected through a graph. Given a d-multi-dimensional dataset S with distance δ (e.g. the Euclidian distance), in which items $s \in S$ are vectors in $\mathbb{R}^d$, a graph $G = (V, E, c)$ can be induced, in which V is the set of vertices, E is the set of edges, and c is the cost function for the edges. In such a graph G:

[1]See section 6 for Internet repositories and random generators of network topologies.

[2]You can find an introduction to MPI in the polycopy.

- vertices $v \in V$ are data items of S, and
- every two vertices u and v are connected through an edge $e_{u,v} \in E$. The cost of such an edge is $c(u, v) = \delta(u, v)$ in $\mathbb{R}^d$.

Given a value k, the MST T over a graph G corresponding to a dataset S allows to determine k clusters by removing the $(k-1)$ most "expensive" (with highest distance) edges in the MST, in a reduced graph $T' \subset T$. By definition, T' consists of k connected components. The set of vertices of each connected component is a cluster of the MST-based clustering.

Task 5 Implement this MST-based k-clustering algorithm, by using the Kruskal algorithm for MST. Compare its complexity and performance with the k-means algorithm, with Forgy initialization (as implemented in TD3), and discuss your results.

5.1 Extensions

The MST clustering algorithm sketched in section 5 takes the number of clusters, k, as input of the algorithm. Sometimes, however, the number of clusters in a dataset is not known in advance and needs to be inferred from the data itself.

Different strategies can be proposed for MST-based clustering without an explicit value for k. One of the simplest ones consists of scanning the range of feasible k's, perform a MST k-clustering for each of them (as implemented in Task 5), and choose the optimal clustering according to a given metric (e.g., smallest ratio between intra-cluster and inter-cluster distance). Others can be found in the literature: see e.g. [**?**].

Task 6 Explore the existing literatures for MST-based clustering strategies without explicit number of cluster (k). Implement one of these strategies and evaluate it against some of the available datasets (see section 6).

6 Graphs and datasets

6.1 Graphs

There are many Internet topology repositories and random graph generators available in the Internet. Among the repositories, different samples of computer network topologies are available in the Internet Topology Zoo project [11]. Among random graph generators, it can be highlighted the Autonomous Supélec Hierarchical Interdomain Infering Program (aSHIIP) [13], a random topology generator developed at Supélec.

For simplicity, it is recommended to use aSHIIP for the evaluation of implemented algorithms. aSHIIP generates easy-to-parse graphs with *no weights* (i.e., if needed, weights will have to be assigned a posteriori) according to several methods: Waxman [5], Erdos-Rényi [8] and Barabási-Albert [3], among others. Some example topologies based on Erdos-Rényi ($p = 0.1$) and Barabási-Albert ($m_0 = 20$, $m = 2$) are provided for $n = \{25, 50, 100, 150\}$ nodes; students are encouraged to consider and discuss also other models.

6.2 Datasets

There are also public datasets that can be used in data science projects, to evaluate techniques, heuristics and strategies. The Springboard blog [16] provides an heterogenous list of such datasets; you are encouraged to look also on your own. For MST-based clustering techniques, these datasets need to be first transformed into graphs, as described in section 5, before usage.

References

[1] T. Pusateri (2003): Distance Vector Multicast Routing Protocol. Internet-Draft. https://tools.ietf.org/html/draft-ietf-idmr-dvmrp-v3-11.

[2] A. Grama, A. Gupta, G. Karypis, V. Humar (2003). Introduction to Parallel Computing, Second Edition. Addison-Wesley, 2003. ISBN 0-201-64865-2.

[3] R. Albert, A.-L. Barabási (2002). Statistical mechanics of complex networks. Reviews of Modern Physics. 74 (1): 4797. ISSN 0034-6861.

[4] W. Guang-rong, G. Nai-jie (2000): An efficient parallel minimum spanning tree algorithm on message passing parallel machine. J. Softw. 11(7), 889898 (2000).

[5] B. M. Waxman (1991): Routing of Multipoint Connections, in Broad-band Switching: Architectures, Protocols, Design, and Analysis, IEEE Comp. Soc. Press, 1991, pp. 34752.

[6] R. G. Gallager, P. A. Humblet, P. M. Spira (1983): A Distributed Algorithm for Minimum-Weight Spanning Trees. ACM Transactions on Programming Languages and Systems, vol. 5, no. 1, Jan. 1983, pp. 66-77.

[7] M. Sollin (1965): Le tracé de canalisation. Programming, Games, and Transportation Networks.

[8] P. Erdos, A. Rényi, (1959): On Random Graphs. I. Publicationes Mathematicae 6: 290297.

[9] R. C. Prim (1957): Shortest connection networks and some generalizations, Bell System Technical Journal, 36 (6): 13891401.

[10] J. B. Kruskal (1956): On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematical Society. 7: 4850.

[11] Internet Topology Zoo: http://topology-zoo.org. University of Adelaide.

[12] Cisco Systems: Spanning Tree Protocol. http://www.cisco.com/c/en/us/tech/lan-switching/spanning-tree-protocol/index.html

[13] aSHIIP, A random topology generator of interdomain: http://wwwdi.supelec.fr/software-orig/ashiip/. École supérieure d'éléctricité (Supélec). (Also available at https://github.com/scalen/wais-model-internet)

[14] Oleksandr Grygorash, Yan Zhou, Zach Jorgensen (2006): Minimum Spanning Tree Based Clustering Algorithms. Proc. 2006 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06).

[15] M. Laszlo, S. Mukherjee (2005): Minimum spanning tree partitioning algorithm for microaggregation. *IEEE Transactions on Knowledge and Data Engineering*, Volume 17, Issue 7, July 2005.

[16] T. J. DeGroat (2018): 19 Free Public Data Sets for Your First Data Science Project. Springboard Blog, post on August 21.