Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 92 (2016) 513 – 519

Procedia Computer Science

2nd International Conference on Intelligent Computing, Communication & Convergence

(ICCC-2016)

Srikanta Patnaik, Editor in Chief

Conference Organized by Interscience Institute of Management and Technology

Bhubaneswar, Odisha, India

An Efficient Greedy Minimum Spanning Tree Algorithm Based on Vertex Associative Cycle Detection Method

Prantik Biswas[a], Mansi Goel[a], Harshita Negi[a], Megha Datta[a] *

[a]National Institute of Technology, Kurukshetra, 136119, India

Abstract

The minimal spanning tree problem is a popular problem of discrete optimization. Numerous algorithms have been developed using the traditional approach but with the emergence of modern-day complex data structures, new algorithms have been proposed which are more complex yet asymptotically efficient. In this paper we present a cycle detection based greedy algorithm, to obtain a minimal spanning tree of a given input weighted undirected graph. The algorithm operates on the idea that every connected graph without any cycle is a tree. At successive iterations, the algorithm selects and tests if the highest degree vertex is a member of any cycle to remove the most expensive edge from the cycle associated with it. The iteration continues until all the cycles are eliminated to obtain the resultant minimal spanning tree. The simplicity of the algorithm makes it easier to understand and implement in any high-level languages. The proposed approach will be beneficial in analyzing certain class of problems in science and engineering.

Peer-review under responsibility of the Organizing Committee of ICCC 2016

Keyword: Discrete Optimization; Minimal Spanning Tree; Greedy Algorithm; Cycle Detection; Graph Algorithm.

* Corresponding author. Tel.: +91-983-643-8182.
E-mail address: pranmasterbi@gmail.com.

Peer-review under responsibility of the Organizing Committee of ICCC 2016
doi:10.1016/j.procs.2016.07.376

1. Introduction

The minimal spanning tree problem (MSTP) is a notable problem of combinatorial optimization. It deals with the problem of obtaining a tree of minimum weight that spans all the vertices of a weighted, undirected and connected graph, where the weight of the tree corresponds to the sum of weights of its edges. It is widely applied in various fields of science and technology ranging from computer and communication networks, knowledge engineering, wiring connections, VLSI circuits design to a large class of optimization problems. Recent approaches in analyzing various biomedical problems like medical imaging, bio-terrorism, etc have made an extensive use of the concepts of minimal spanning tree (MST). In fact recent advances in clustering algorithms also deploy the concepts of MST.

Numerous systematized solution techniques exist for solving the MSTP. One of the first known solutions was given by Boruvka [11] in 1930. Two most popular MST algorithms are due to Kruskal [12] and Prim [13]. These three algorithms are often referred to as the classical algorithms for solving the MSTP.

The rich history of the MSTP is also well documented. Pierce [1] analyzed the details of all the classical algorithms related to the minimal spanning tree problem. Maffioli's [2] survey stressed on the asymptotical complexity of the methods used to solve different types of optimum undirected tree problem. The survey by Graham and Hell [3] gives an insight to the algorithmic technique for solving the minimal spanning tree problem, even tracing their independent sources/origin.

Researchers in recent past have focused on devising computationally faster algorithms. With the emergence of modern data structures and improved hardware support, efficient implementation techniques were also invented aiming to speed up these classical algorithms. In 1984, Haymond, Jarvis and Shier [5] elaborated various computational methods for MST algorithms. Non-greedy approaches for MSTP were also proposed [6]. A detailed survey of different computational experiments is also available [7]. Finally the first linear expected-time randomized, recursive algorithm for the MSTP was proposed by Karger [4] suitable for computational models of restricted random access type.

With the advent of parallel computing, different authors began to focus on parallelizing the so called classical algorithms. In 2014 Lončar [18] proposed a technique to parallelize the classical MST algorithms using distributed memory architecture. Osipov [19] presented the Filter-Kruskal algorithm that avoids sorting of edges that are obviously not in the MST. It also provided an equivalent parallelization best fitted for modern multi-core machines. Bader [20] gave several parallel minimal spanning tree algorithms (three variations of Boruvka and a new version) that can be easily implemented on irregular-structured graphs.

With the advent of parallel computing, different authors began to focus on parallelizing the so called classical algorithms. In 2014 Lončar [18] proposed a technique to parallelize the classical MST algorithms using distributed memory architecture. Osipov [19] presented the Filter-Kruskal algorithm that avoids sorting of edges that are obviously not in the MST. It also provided an equivalent parallelization best fitted for modern multi-core machines. Bader [20] gave several parallel minimal spanning tree algorithms (three variations of Boruvka and a new version) that can be easily implemented on irregular-structured graphs.

In the next two sections, we present the problem definition and the general solution techniques. In section 4, we describe our proposed algorithm that can efficiently find the MST. In section 5, we illustrate the working of the proposed algorithm followed by drawing a conclusion in section 6.

2. Problem Definition

A spanning tree of a connected undirected graph G = (V, E), is defined as a tree T consisting of all the vertices of the graph G. If the graph G is disconnected then every connected component will have a spanning tree T_i, where 'i' is the number of connected component, the collection of which forms the spanning forest of

the graph G. A graph may have many spanning trees. If we associate a weight w_i, against each edge of the graph, then the spanning trees may be of varying weights. Among them, the minimum weighted spanning tree, often referred to as the Minimum Spanning Tree (MST), is of general importance. It has a wide variety of application in various fields of science and technology.

An equivalent definition is due to Bazlamacci [23] that states that given an undirected graph G=(V,E), where V denotes the set of vertices with n=$\left|V\right|$ and E the set of edges with m=$\left|E\right|$ and a real number w_i=w(e) for each edge e ε E called the weight of edge e, the MSTP is formally defined as finding a spanning tree T^* on G, such that $w(T^*) = \min_T \sum_{e\in T} w(e)$is the minimum taken over all possible spanning trees of G.

The following lemma forms the basis our proposed algorithm.

Lemma: Every connected graph without any circuit is a tree.

3. General Solutions

Traditional approaches for solving the minimal spanning tree problem are usually greedy in nature. Majority of the classical algorithms build the MST edge-wise, adding the appropriate small edge and excluding the larger ones. The greedy nature of the algorithms is reflected from the fact that the algorithms choose, the best possible edge for insertion into the MST that do not produce a cycle in the sub-graph constructed so far, at every stage or for deletion from the MST, keeping the graph connected.

Well known classical algorithms like Boruvka [11] and Kruskal [12] the disjoint set union algorithm for computing the MST. Prim[13] implemented it with binary heap, d-heap and F-heap. Yao[14], Cheriton [15], Fredman [16] gave new improvements of these classical methods. Tarjan [10] figured out the MST construction process as one of the edge coloring model. Karger [17, 4] proposed the first known almost linear time algorithm with the help of randomized algorithms and recursion. A good survey of all these methodologies can be found in [23].

Modern authors have also concentrated on parallelizing these algorithms. Dalal [24], Srimani [25] gave some elegant solutions. Loncar [18] also gave two new approaches that targets message passing parallel machines with distributed memory. Bader [20] gave a method to compute the minimum spanning forest of a sparse graph using shared-memory.

4. Proposed Algorithm

Our proposed algorithm is based on vertex associative cycle detection algorithm. Various elegant cycle detection algorithm of almost linear order can be easily found [21, 22]. We select the vertex u, having maximum degree from the set of unmarked vertices (V-S), where S is the set of marked vertices. We then check if the chosen vertex is a member of a cycle/knot. If it is so, we then identify the most expensive edge e_x (i.e. the edge with maximum edge weight) and delete it from the cycle as well as from the list of edges E. If a tie occurs, we choose the edge whose terminal vertices have maximum degree. We repeat this process with the selected vertex u until the vertex becomes free of all the cycles it is involved with. Whenever a node becomes free of all the cycles it was associated with, we add it to the set of marked vertices S. We the select another vertex and continue the process. The algorithm terminates when the number of edges |E| becomes equal to |V|-1, since a tree with |V| vertices contains |V|-1 edges. The algorithm is given below.

```
INPUT: A weighted undirected graph G= (V, E).
OUTPUT: A MST T of G.
MST_Algo (G, V, E, T, S)
{
   S=Φ;
   T=G;
   while(|E| > |V|-1)
  {
      SELECT a vertex u∈(V-S) whose degree is maximum
      while (TRUE)
       {
           if (u is a member of a cycle/knot in T)
           {
             SELECT the most expensive edge e_x from the  cycle/knot containing the vertex u from T;
             if(a tie occurs)
             {
                    SELECT the edge e_x whose terminal vertices have maximum degree;
             }
         Remove the most expensive edge e_x from the cycle/knot containing the vertex u from T;
          E = E-e_x;
           }
           else
           {
                S = S ∪ u ;
                 break;
           }
        }
    }
 return T;
```

The proposed algorithm chooses the vertex with maximum degree at every step. The motivation behind this idea is that the probability that a vertex is a member of a cycle increases with its degree. More the degree of a vertex more is the probability that the vertex is associated with some cycle. Moreover tie while eliminating an

edge is determined based on the degree of the terminal vertices of the edges. The greedy nature of the algorithm is reflected from the fact that at every stage the vertex with maximum degree is chosen for checking the cycle associativity.

5. Results and discussions

This section illustrates the execution of the algorithm applied to the input graph shown in figure 1. The algorithm first selects node B because it has highest degree. It then detects that the node B is associated with the cycle B-A-E-B. Next it chooses the most expensive edge and finds that there is a tie, namely AE and BE. However the terminal vertices of BE has maximum degree and hence BE is eliminated as shown in figure 2 by the dotted red lines.

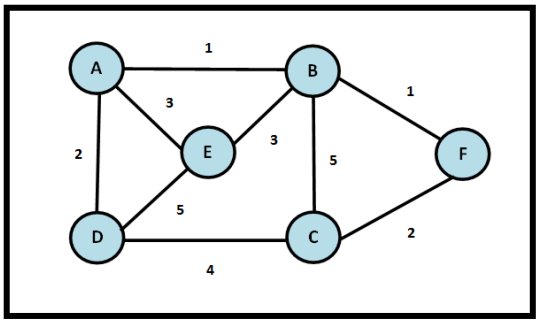

Figure 1: Input Graph

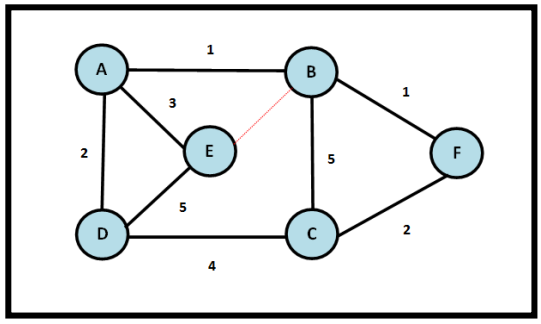

Figure 2: BE deleted

Now the algorithm detects that B is also a member of the cycle B-C-F-B and removes the most expensive edge BC from it (see Fig. 3). Next the cycle B-A-D-C-F-B is detected and DC is deleted (see Fig. 4). At this point the node B becomes free of all cycles and hence vertex B is added to the list of marked vertices S.

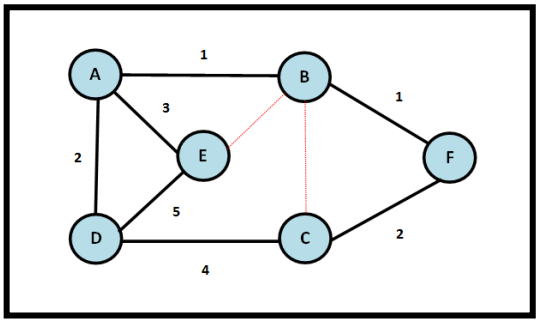

Figure 3: BC deleted

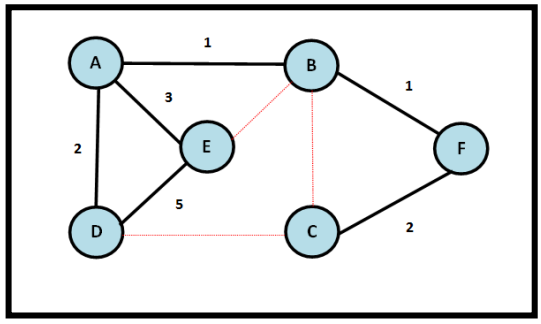

Figure 4: DC deleted

The next vertex chosen for testing is A. The node A is associated with the cycle A-D-E-A and hence De being the most expensive edge is removed as depicted in figure 5. Next A is added to the list of marked vertices.

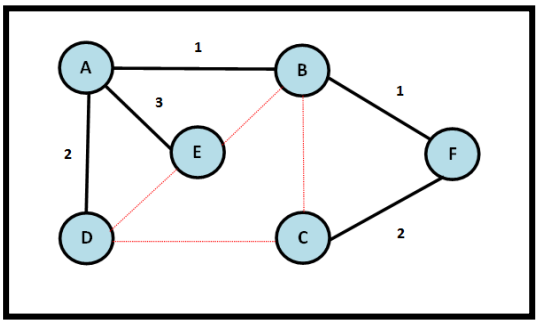

Figure 5: DE deleted

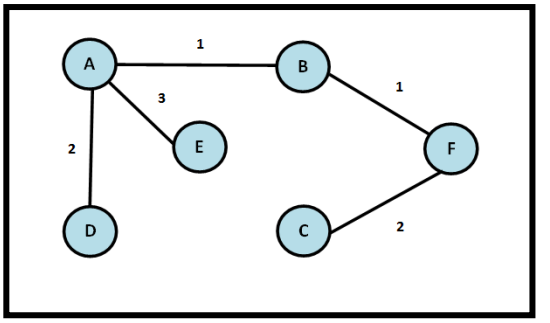

Figure 6: Minimal Spanning Tree

At this iteration the number of edges equals one less than the number of vertices and hence the algorithm terminates returning the resulting minimal spanning tree as shown in figure 6.

One advantage of this method is that it eliminates the need to sort the edges. Additionally in most of the cases, removal of an edge causes removal of multiple cycles. For example removal of BC in figure 3 eliminates four cycles from the graph. The set of marked vertices S ensures that the selection of vertices is not repeated.

6. Conclusion

In our work we have given an algorithm that can be easily implemented on a single machine. It can be further extended for distributed systems (distributed graphs) by applying various parallel computing techniques. A major advantage of this algorithm is that it expels the need of sorting the edges of the weighted graph. Additionally removing one of the edges from a selected cycle also breaks multiple cycles there by making it even faster. Thus we can conclude that it is an efficient methodology for finding the minimal spanning tree of a given weighted undirected graph.

References

[1] Pierce A R. Bibliography on algorithms for shortest path, shortest spanning tree and related circuit routing problems (1956-1974). *Networks*. 1975; **5**:129-49.

[2] Maffioli F. Complexity of optimum undirected tree problems: a survey of recent results. *Analysis and design of algorithms in combinatorial optimization*. *CISM Courses and Lecture*. New York: Springer .1981; **266**: 107-28.

[3] Graham R L, Hell P. On the history of the minimum spanning tree problem. *Annals of the History of Computing*. 1985; **7**:43-57.

[4] Karger D R, Klein P N, Tarjan R E. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the Association for Computing Machinery*. 1995; **42/2**:321-8.

[5] Haymond R E, Jarvis J P, Shier D R. Computational methods for minimum spanning tree algorithms. *SIAM Journal on Scientific and Statistical Computing*. 1984; **5/1**:157-74.

[6] Glover F, Klingman D, Krishnan R, Padman A. An in-depth empirical investigation of non-greedy approaches for the minimum spanning tree problem. *European Journal of Operational Research*. 1992; **56**:343-56.

[7] Moret B M E, Shapiro D. How to find a minimum spanning tree in practice. *New results and new trends in computer science: proceedings, Graz, Austria*, *June 1991*. Lecture Notes in Computer Science, Vol. 555. Berlin: Springer, 1991. p. 192-203.

[8] Balakrishnan VK. *Network optimization*. London: Chapman and Hall, 1995.

[9] Gondran M, Minoux M. *Graphs and Algorithms*. New York: Wiley, 1984.
[10] Tarjan R E. Data structures and network algorithms. *CBMS-NFS Regional Conference Series in Applied Mathematics. Philadelphia*: Society for Industrial and Applied Mathematics, 1983. p. 44.
[11] Boruvka O. O jistém problému minimálním. *Práca Moravské Přírodovědecké Společnosti (in Czech)*. 1926; **3**:37-58.
[12] Kruskal J B. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proceedings of the American Mathematical Society* 1956; **7**:48-50.
[13] Prim R C. Shortest connection networks and some generalizations. *Bell Systems Technical Journal*. 1957; **4**:53-7.
[14] Yao A. An O(|E| log log |V|) algorithm for finding minimum spanning trees. *Information Processing Letters*. 1975; **4**:21-3.
[15] Cheriton D, Tarjan R E. Finding minimum spanning trees. *SIAM Journal on Computing*. 1976; **5**:724-42.
[16] Fredman M L, Tarjan R E. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the Association of for Computing Machinery* 1987; **34/3**:596-615.
[17] Karger D R. Random sampling in matroids, with application to graph connectivity and minimum spanning trees. *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science. Los Alamitos, California*: IEEE Computer Society Press, 1993. pp. 84-93.
[18] Lončar V, Škrbić S, Balaž A. Parallelization of minimum spanning tree algorithms using distributed memory architectures. *Transactions on Engineering Technologies*, DOI: 10.1007/978-94-017-8832-8_39, Springer Science+Business Media Dordrecht 2014.
[19] Osipov V, Sanders P, Singler J. The Filter-Kruskal minimum spanning tree algorithm. *Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments,* By: SIAM 2009: pp. 52-61.
[20] Bader A, Cong G. Fast shared-memory algorithms for computing the minimum spanning forest of sparse graphs. *Journal of Parallel and Distributed Computing*. 2006;1366-78. doi:10.1016/j.jpdc.2006.06.001.
[21] Boukerche A, Tropper C. A distributed graph algorithm for the detection of local cycles and knots. *IEEE Transactions on Parallel and Distributed Systems*. 1998; **9**: 748-57.
[22] Manivannan D, Singhal M. An efficient distributed algorithm for detection of knots and cycles in a distributed graph. *IEEE Transactions on Parallel and Distributed Systems.* 2003; **14**: 961-72.
[23] Bazlamacci C F, Hindi K S. Minimum-weight spanning tree algorithms a survey and empirical study. *Computers and Operation Research*. 2001; **28**: 767-85.
[24] Dalal Y K. A distributed algorithm for constructing minimal spanning tree. *IEEE Transactions on Software Engineering* 1987; **13**: 398-405.
[25] Srimani P K, Xu Z. Self-stablizing algorithms of constructing spanning tree and weekly connected minimal dominating set. *27th International Conference on Distributed Computing Systems Workshops* 2007.