ISBN 978-981-11-0008-6
Proceedings of 2016 6th International Workshop on Computer Science and Engineering (WCSE 2016)
Tokyo, Japan, 17-19 June, 2016, pp. 95-102
doi: 10.18178/wcse.2016.06.017

Connected Dominated Set Based Node Centrality in Wireless Networks

T. Virajita Sharma, Usha Kiran Singh and Debasis Das [+]

Department of Computer Science and Engineering, NIIT University, Neemrana, Rajasthan –301705, India

Abstract. The number of people using wireless devices and internet has increased remarkably in the previous decade, and so is the need to route/broadcast information to the entire network. One of the latest and most promising approaches is finding a Connected Dominating Set (CDS) in network. Another popular concept in the social network analysis is Node centrality, because of its ability to find popular/influential nodes. In this paper, we have formulated a simple but efficient algorithm, which first finds a CDS in the network, and then with the help of node centrality metrics it finds a central node, for more effective routing/broadcasting. We have increased simplicity by making changes in the existing algorithms, incorporated Node Centrality, and ensured its efficiency by a step-by-step complexity analysis and constant changes and improvements.

Keywords: broadcasting, CDS, central node, connected dominating set, degree centrality, eigenvector centrality, routing, and wireless networks.

1. Introduction

Due to the technological advancements, and the growing popularity and efficiency of wireless devices, there is a surge of research going on, on wireless and wireless ad hoc networks in this 21st century. The speciality and the reason for it being a revolution, is of course the sense of repose it offers to its users, as there is no wired connection to maintain every second. But as we know, these wireless networks [15] are not as efficient as wired connections, due to low bandwidth, varying signal strength due to mobility, and the various obstructions between the router and the device which cause a high delay in transmission, and the changing topology and the uncertain lifetime of the link in the case of ad hoc networks. So researchers across the world are working hard, aiming to make routing more effective and efficient in these scenarios.

Routing is a process where the packets containing information have to be sent, from the source, to the destination, along the path with minimum hops, with low end-to-end latency and overhead. But broadcasting a message does not always ensure full efficiency. There was flooding [1] that did this, and then came broadcast routing [2]. But one of the interesting methods that uses gateway hosts to form a dominating set, is Connected Dominated Set (CDS)-based routing [3], [4], [8], [9], [13]. Every node in the network is either in the subset called Dominating Set, or is a one-hop-neighbour to a node in the dominating set. The small connected dominating set is used as a backbone for communications, and nodes that are not in this set communicate by passing messages through neighbours that are in the set. This greatly simplifies the routing process [3], and broadcasting information, because only the nodes in the dominating set need to keep the routing information. The solution of finding whether there exists a minimum CDS of the size less than, or equal, to the given threshold, is NP-complete. But fortunately, researchers along these many years have proposed a number of approximation algorithms, which we will certainly explore.

Node centrality [7], [10] metrics, on the other hand, are quite famous among the researchers studying social network analysis. The centrality metrics calculate and identify one or more most influential nodes in

[+] Corresponding author. Tel.: +91 7790844892.
E-mail address: Debasis.Das@niituniversity.in; deba16@gmail.com.

the network. ‘Influence’ here can be seen as the importance of the node in the network, such as having maximum connections, located in the centre of the network so that it can reach out to many other nodes, etc. It was initially used to measure popularity of that node in the social networks. There are many different node centrality metrics such as Degree centrality, Closeness centrality, Betweenness centrality, Eigenvector centrality, and Katz centrality, to name a few. We are going to use the connected dominating set based routing, along with Eigenvector and Degree centrality, in our approach.

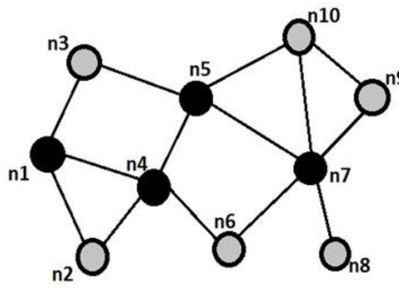

Fig. 1: A sample graph showing a Connected Dominating Set. The nodes in black are the nodes in CDS, i.e., CDS= {n1, n4, n5, n7}.

2. Related Work

As the world is digitizing rapidly, and staying connected to the world has become a basic necessity, faster and effective ways of broadcasting information can prove to be very useful, in the days to come. Flooding was the initial idea of broadcasting, but was not that efficient as the information was blindly passed on to the neighbour. Later more efficient and new approaches came up [1]. The author proposed two new flooding methods, to improve the classical flooding, and a multicast tree, for multicasting a message. The author of [2] compares all the broadcast routing algorithms, to find the most power-efficient routing algorithm among them.

But we thought the CDS-based routing would be more feasible and efficient for broadcasting, for a number of reasons, and a lot of existing researches of some elite authors have helped us a lot. For example, the authors of [8] made a list of all the algorithms that are CDS-based. They presented a detailed analysis of each algorithm and ordered the list from the simplest to the most complex algorithms, making it extremely reader friendly. It helped us a lot in choosing the right algorithm for our needs, among so many. Then there is [3], where the authors formulated their own simple and distributed algorithm, for CDS-based routing. This work was the main inspiration for our work. They made sure that the CDS is connected and close to minimum, of course because exact minimum CDS is NP-complete, as mentioned earlier. Since their aim was to use their approach for the entire routing process as such, they included all intermediate nodes of any shortest path, in their dominating set. So while routing, the all-pair shortest path algorithm will be applied only to the dominating set. They marked the nodes having maximum connections as T, and the unmarked as F, after considering and comparing the degree of each node. They simulated and produced a final algorithm of complexity $O(\Delta^2)$, where Δ is the maximum node degree in the graph/network, which looked very attractive and promising to us.

We continued to look for more recent works, and came across [9], where the authors incorporated clustering in their algorithm of CDS, applying it in MANETs. They applied the algorithm on real datasets, and proved their addition to the earlier simpler algorithm will work more efficiently if there are a large number of nodes, like in MANETs. In [4], on the other hand, the authors have classified all the existing algorithms that make a CDS, into categories based on the approach. They surveyed each algorithm, and calculated the overall time and message complexity. They concluded the paper with the reasons on why making a CDS before routing/broadcasting [15] can be beneficial for future systems and Wireless Sensor Networks, such as, reducing communication overhead, increasing network effective lifetime, etc, which strengthened our choice of CDS.

So for making our own algorithm, we planned to incorporate a centrality metric along with the CDS, in our paper. Centrality metrics, in general, are used in social network analysis. The book by the authors of [10], guided us very much, by giving the simplest definitions of centrality as a measure of the importance of a node in a network, and all the different metrics. The basic/simplest one of all the centrality metrics is the Degree centrality [10], [12]. Degree of a node is simply the total number of adjacent edges of that node, represented by the formula,

$$C_d(v_i) = D_i$$

where the subscript *d* is for degree, *v* stands for vertices, *i* is the node we are calculating for, and *D* is the no. of one-hop neighbours or the adjacent edges as mentioned above. So this measure can be seen as the number of connections of that node. But the result can be a huge number if we work with large networks, so the degree values are normalized [10] by dividing it with (n-1), which gives us:

$$C_d(v_i) = (D_i/(n-1))$$

where *n* is the total number of nodes in the network, and the other symbols same as above. But this alone cannot determine the overall popularity, because what if the node indeed has many connections, but is located at the corner of the network as a whole. Then we cannot assure that this node will reach out to maximum people. This is when Closeness centrality and Betweenness centrality became popular.

Both betweenness and closeness centrality metrics rely on identifying and computing the lengths of shortest paths in the networks. Betweenness centrality looks into the situation, in a slightly different perspective. It measures the importance of the node, by computing how important the node is, in connecting other nodes. So the importance of a node v_i, is the number of shortest paths between other nodes that pass through v_i.

$$C_b(v_i) = \sigma_{st}(v_i)/\sigma_{st}$$

where the subscript *b* stands for betweennes, *i* is the node index, σ_{st} is the number of shortest paths between one node *s* and another node *t*, and $\sigma_{st}(v_i)$ is the no. of shortest paths from the node *s* to *t*, but passing through the node v_i. Closeness centrality, on the other hand, believes that more central the nodes are, more quickly they can reach other nodes. It is defined as,

$$C_c(v_i) = 1/\, l_{v_i}$$

where $l_{v_i} = (1/(n-1)) \sum_{v_i \neq v_j} \sigma_{i,j}$ is the average shortest path length of v_i to other nodes. The subscript c is for closeness, *i* and *j* are the node indices, $\sigma_{i,j}$ is the distance between nodes *i* and *j*, i.e., no. of hops.

But we chose Eigenvector centrality [10], [5], because we wanted to totally decrease the probability of two nodes having the same centrality value. There are two ways of calculating Eigenvector centrality, with the help of eigenvector matrix and eigenvalues (Perron-frobenius Theorem [10], [5]), or coding the general logic. Eigenvector centrality generalizes the degree centrality, by giving importance to the no. of neighbors. It believes that having more important neighbors is more significant than having just more neighbors. So here we calculate the degree of each neighbor, then the degree of each of those neighbors, and so on, till the end, in a recursive fashion. This will help us to observe which nodes will reach out to the maximum nodes, more accurately and effectively. The formula for eigenvector centrality can be seen as: $x_v = \sum_{t \in G} a_{v,t} x_t$, where $a_{v,t}$ is the adjacency matrix, value 1 if a link exists, 0 otherwise, and x_v is the eigenvector centrality value of node *v*. In [11], the authors have sampled the networks, and computed the stability of each centrality metric, helping us decide among so many centrality metrics clearly. They have found out that Eigenvector centrality shows most stability in the networks, even after sampling. They concluded that In-degree was the most stable even in low samples, and Eigenvector centrality as a raw score was stable, because it was able to capture the network structure effectively. We even studied some other Eigenvector-like centrality measures with the help of the authors of [6]. And the authors of [5] gave a perfect justification, and a detailed explanation of various applications of Eigenvector centrality. All these works helped us to choose this metric.

A more recent research that motivated our work was [13]. The authors of [13] have formulated a unique CDS algorithm that works in Unit Disk graphs. In geometric graph theory, the unit disk graph is a graph with one vertex for each disk in the family, and with an edge between two vertices whenever the corresponding vertices lie within a unit distance of each other. They chose UD graphs, because it is the most suitable model for wireless sensor networks, which happens to be their main application. Their detailed step-by-step algorithm efficiently finds a CDS. The main reason for choosing the connected dominating set remains the same, that it helps reduce interference between nodes, and also energy transmission. Another recent research that moved us regarding centrality metrics is [14]. They have introduced a node prominence profile, to predict node degree centrality. They have taken up the issue that how over the time dynamic networks evolve, and node centrality of nodes change. They have explained how a node may become important over time, which may be a result of its individual achievement or neighbourhood structure, which represent aspects of preferential attachment and triadic closure respectively. Their new method Node Prominence Profile effectively uses triadic closure, to specify the forbidden positions, with the help of various statistics. It moved us towards selecting Degree centrality. The following sections describe our approach, motivated from the above researches.

3. Problem Statement

We have seen that despite finding a minimum connected dominating set is NP-complete, a bunch of approximation algorithms have been formulated, and still counting. But the approximation algorithm we chose still has some extra components that could be avoided. Removing certain components can benefit us by reducing the overall complexity, increasing the effectiveness of the algorithm, which is one of the most important goals. And also, when dealing with degree centrality, the simplest node centrality metrics, there are high chances for two (or more) nodes to have the same value, which can be a problem too. In this paper we have solved these with our simple algorithm.

4. Objective

We have seen how calculating a CDS can prove to be more effective for broadcasting information. We have also seen which centrality metrics are more effective than the others. In this paper, our objective is to effectively and efficiently broadcast information in a network. Most of the existing algorithms use spanning tree, which we have avoided, and still succeeded. The existing algorithms only find a CDS in the network. But we, after finding the CDS, are going to find one central node in the CDS. The information can be given to the central node initially, and then it will reach all the others. We are going to find the central node on the basis of maximum degree. Since degree is the simplest centrality metric, there can arise a situation where two nodes can have the same degree. We are going to have a provision to see which node is more central among the conflicting nodes, which has never been done before.

5. Proposed Approach

As mentioned earlier, we will calculate CDS of the network. The algorithm we chose will be discussed further in this section. While choosing the central node among the CDS, if two or more nodes have the same degree, then we will calculate Eigenvector centrality of those nodes, to see who is more central. So to summarize, our algorithm will:

a) Find a CDS of the network.
b) Find a central node in the CDS using Degree centrality.
c) If two (or more) nodes of CDS have same degree, we calculate their eigenvector centrality and finalise the central node as the one with highest eigenvalue.

The algorithm we are going to use, uses graph coloring ([4], Algorithm I in [8]). The detailed comparison with other existing works, is given at the end of this section. The CDS algorithm goes like this:

1. All nodes are initially colored white.
2. The starting node is colored black. All the neighbors of the black node are colored gray.
3. Select a gray node that has the maximum white neighbors and color it black. Its white neighbors are marked gray.
4. Repeat step [3] until there are no more white nodes.

This simple algorithm given above will give us the CDS in the form of black nodes. The figure is an example network, on which this algorithm is applied. We can observe the CDS, i.e., the black nodes, nodes

A, B, C, D and E. The rest of the pseudocode, along with finding the central node is as given below.

5.1. Algorithm: CDS and Node Centrality Based Central Node Selection

Input: Wireless connected graph G

Objective: Find a CDS in G, and then find a central node in the CDS using Node Centrality

Output: a Central Node in the CDS

```
CentralNode()
{    FindCDS(G)
     Central_Node = maxDegree(CDS[]) }
FindCDS(G)
{     All vertices v = white and X = black
      Every one-hop neighbour of X = grey
      REPEAT
          (X = grey node) AND (If X has max white neighbours)
                then X = black AND every one-hop neighbour of x= grey
      UNTIL no white nodes left }
maxDegree(CDS[])
{     For all vertices v in CDS[]
            degree[v]= all one-hop neighbours
      For all vertices v in CDS[]
            If (degree[x]==degree[y])  then A[i]=M
      If (A[]==φ)  then M = max(degree[])
      Else Evc(A[])
      return M  }
Evc(A[])
{     V = adjacency matrix of all nodes in A
      w[]=eigenvector matrix
      For each node i in the array A[]
            For each neighbour j of i
                  w[i] = Σ v[i][j]*w[j]
      S = sum of the entries of w.
      Divide each entry of w by S.  //normalization
      Return max(w[])  }
```

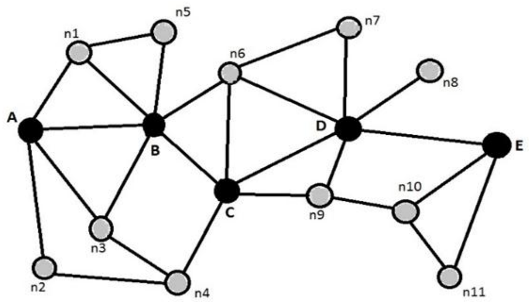

Fig. 2: A sample graph, after applying CDS based central node selection algorithm. The black nodes denote the Connected Dominating Set.

The first function is CentralNode(), which internally calls two other functions FindCDS(), which finds the CDS in the given graph, and maxDegree(), which finds the central node in the CDS. Inside FindCDS(), the resulting set of black nodes is stored in the array CDS[], so that it can be used further. The array found will then be passed to maxDegree() for further processing. Inside maxDegree(), the degree centrality values of each nodes in the CDS[] are compared. Finally if any degree values come out to be equal, all those conflicting nodes are stored in a separate array A[], and then passed to Evc() for further processing. In Evc(), the eigenvector centrality of the conflicting nodes is calculated, highest value corresponding to the more central node. As mentioned earlier, degree of each neighbour of the conflicting node is calculated recursively, to decide which node can reach out to maximum nodes.

Fig. 2 shows a sample graph, on which the above mentioned algorithm has been applied. We can see the nodes in the Connected Dominating Set as the nodes coloured in black. Now, to calculate the Central Node, first the Degree centrality of all the nodes in CDS are compared. That results us with one conflicting pair of nodes, node B and node D. So according to the algorithm, when we apply Eigenvector centrality and add the degree of each neighbour, we can easily conclude that node D is the Central Node.

6. Complexity Analysis

If formulating an algorithm is an important step, measuring the effectiveness of the algorithm is equally important. To do the latter, we have performed a step-by-step complexity analysis as follows.

6.1. Step-by-Step Complexity Analysis

The algorithm is one of the simplest, as seen above, and hence the complexity too. We have calculated the complexity step-by-step:

a) The function FindCDS() will have a best case complexity of $2 \ln \Delta$, as proved in [3], where Δ can be seen as the maximum degree of a node in the network. The worst case complexity of the function FindCDS() would be O(V), where V is the total no. of nodes in the network. This is because, on the worst case scenario, the network can be seen as a straight line, with nodes like beads in a necklace. In that case, the function FindCDS() would consider every node to be in CDS, because that will be the solution. Hence O(V).

b) The best case of the function Evc(),would of course be when only one match occurs while finding maxDegree(). So the best case complexity of Evc() should be O(2Δ), where Δ is the maximum number of nodes that can be reached from a node, as specified in the algorithm. And the worst case would be O(VΔ), where V is the number of matches that is possible at the maximum, i.e., if all the nodes in the network are in CDS (topology like beads in a necklace), and every node has the same degree as the other. And Δ of course the maximum no. of nodes reached from a node, as above.

c) Now coming to the function maxDegree(), the best case is when no match will occur, i.e., O(2v), where v is the total nodes is CDS. The worst case of maxDegree() will include Evc(), as it is calling Evc() internally. So worst case complexity of maxDegree() must be O(V+ VΔ) ~ O(VΔ). This can go up to V^2 only when each node is connected to every other node, which is quite imaginary in real life netwoks.

d) Hence, the best case complexity of the entire algorithm: O(2V), worst case: O(VΔ).

6.2. One Alternative Way

Instead of using the function maxDegree() in between, the Evc() can be run on the entire CDS too, the maximum value corresponding to the more central node. This can be done if we are very particular on finding the most central node at any cost, not even the nearest to most. But it has been observed that this will affect the best case complexity, i.e., the algorithm will give O(VΔ) every time, as we will run Evc() every time, on the entire FindCDS(). Other than this no drawback would be observed. So if the system is fine with the time complexity, and wants the most central node, then this can be implemented. But it will be advised not to implement this method even if one of these two conditions are not required in your scenario, because of course, if your system has to be fastest, this will be a waste of time, and if you are not that particular for the central node, then also this will be a waste of time.

6.3. Comparison

As mentioned earlier, it is NP-complete to test whether there exists a connected dominating set with size less than a given threshold, and hence we have seen the different approximation algorithms for finding a connected dominating set, without the threshold [4, 3, 1, 10, 14]. Our algorithm to find CDS is similar to the first algorithm of [8]. This algorithm is the base of many other's work [3, 1, 14]. The algorithm is simple, and uses graph colouring. But it has been observed that all the existing versions of this algorithm make, and use a spanning tree of the network, which can be a drawback if we consider the computational complexity. We have planned to avoid making the spanning tree, which, according to observations, will decrease an approximate worst case complexity of $O(V^2)$ and best case complexity of $O(ElogV)$. So, avoiding the spanning tree, will prove to be quite useful, as seen in our algorithm.

All the algorithms we have seen, use CDS [1, 3, 4, 10, 14], or the centrality metrics [6, 13, 15], separately. No existing algorithm has incorporated both of these techniques together, for a better algorithmic approach. Moreover, unlike our algorithm, nobody else has the provision to find the central node in CDS. Finding the central node in CDS further simplifies the routing/broadcasting process, as the information would only have to be given to the central node. In this way, our algorithm is quite different and unique than all the other algorithms we have seen.

7. Conclusion and Future Work

In this paper, we have proposed an algorithm to effectively broadcast information in any type of wireless network, in a simple and efficient way. Our algorithm first calculates the connected dominating set in the wireless network, and then finds a central node amongst the CDS nodes, so that while broadcasting information will first be given to the central node, and then will be passed on to the rest of the wireless network. Unlike existing CDS algorithms, we have eliminated the use of a spanning tree while calculating CDS, reducing the overall complexity. Even while finding the central node, we are not taking a random node, but a most central node that can reach out to many nodes (maximum degree), making it efficient for broadcasting information. The overall time complexity of our algorithm will lie between O(2V) and O(VΔ), depending on the network.

The future work related to this algorithm will extend to simulating and testing the algorithm in a real world data set. More efforts on how to make the algorithm more efficient by decreasing the complexity further, will be made. We are going to apply this concept of ours in wireless ad hoc networks, or wireless sensor networks, and check its performance.

8. References

[1] H. Lim, and C. Kim. Multicast tree Construction and Flooding in Wireless Ad Hoc Networks. *3rd ACM international workshop*. Modeling, analysis and simulation of wireless and mobile systems, Seoul National University, Korea. 2000, pp. 61-68.

[2] I. Kang, and R. Poovendran. A Comparison of Power-Efficient Broadcast Routing Algorithms. *Global Telecommunications Conference*. University of Washington, Seattle. 2003, pp. 387 – 392.

[3] J. Wu, and H. Li. On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks. DIALM '99, *3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. Florida Atlantic University, Florida. 1999, pp. 7-14.

[4] Asgarnezhad, R., & Torkestani. Connected dominating set problem and its application to wireless sensor networks. In *The First International Conference on Advanced Communications and Computation*. INFOCOMP.

[5] L. Spizzirri. Justification and application of eigenvector centrality. *Algebra in Geography: Eigenvectors of Network*. Vol. 53, No. 5, 1980, pp. 269-276.

[6] Bonacich, P., & Lloyd, P. Eigenvector-like measures of centrality for asymmetric relations. *Social networks.* 23(3), pp. 191-201.

[7] Amir Noori. On the Relation between Centrality Measures and Consensus Algorithms. *International Conference on High Performance Computing and Simulation*. Islamic Azad University, Iran. 2011, pp. 225-232.

[8] Sudipto Guhay, and Samir Khullerz. Approximation Algorithms for Connected Dominating Sets. *Proceedings of the Fourth Annual European Symposium on Algorithms.* University of Mayland, College Park. ESA'96, pp. 179-193.

[9] D. Cokuslu, K. Erciyes, and O. Dagdeviren. A Dominating Set Based Clustering Algorithm for Mobile Ad hoc Networks. *6th International Conference in Computational Science*. Izmir Institute of Technology, 2006, pp. 571-578.

[10] R. Zafarani, M. Abbasi, and H. Liu. Network Measures, ch. 3. *Social Media Mining: An Introduction*, Cambridge University Press, 2014.

[11] E. Costenbader, and T. Valente. The stability of centrality measures when networks are sampled. *Social Networks*. Vol.25, 2003, pp. 283–307.

[12] T. Opsahla, F. Agneessensb, and J. Skvoretzc, Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*. Vol.32, 2010, pp. 245-251.

[13] G. Purohit, and U. Sharma. Constructing Minimum Connected Dominating Set: Algorithmic Approach. *International journal on applications of graph theory in wireless ad hoc networks and sensor networks*. GRAPH-HOC, Vol.2, 2010.

[14] Y. Yang, Y. Dong, and N. Chawla. Predicting Node Degree Centrality with the Node Prominence Profile. *Scientific Reports*. Vol.4, 2014.

[15] D. Das, R. Misra and A. Raj. Approximating Geographic Routing using Coverage Tree Heuristics for Wireless Network, *Wireless Networks*. Springer US, Vol. 21, No. 4, 2015, pp. 1109-1118.