STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LXV**, Number 1, 2020
DOI: 10.24193/subbi.2020.1.05

PERFORMANCE EVALUATION OF BETWEENNESS CENTRALITY USING CLUSTERING METHODS

BENCE SZABARI AND ATTILA KISS

Abstract. Betweenness centrality measure is used as a general measure of centrality, which can be applied in many scientific fields like social networks, biological networks, telecommunication networks or even in any area that can be well modelled using complex networks where it is important to identify more influential nodes. In this paper, we propose using different clustering algorithms to improve the computation of betweenness centrality over large networks. The experiments show how to achieve faster evaluation without altering the overall computational complexity.

1. Introduction

Nowadays, large networks play a vital role in many scientific areas like computer science [17], social engineering [11], biology [13], chemistry [8], [9], building sensor networks [1]. Over the years many centrality measures were defined such as closeness centrality which is based on the average length of the shortest path between a selected node and all other nodes in the graph. [3] The eigenvector centrality where the score of a node is influenced by the scores of its adjacents nodes. [19] Furthermore, betweenness centrality which quantifies the number of times when a node behaves like a connecting bridge along the shortest path between two vertices. [7].

All of these are used to identify the most important nodes within a graph, based on different conditions and criteria. In this paper, we will focus on betweenness centrality. This centrality measure is still quite popular since it can be applied in many fields. A common example is, in a telecommunications network we can find those nodes which have the most influence over the network since a node with higher betweenness centrality has more control and

Received by the editors: 17 May 2020.

2010 *Mathematics Subject Classification.* 68U99, 94C15.

1998 *CR Categories and Descriptors.* C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Network communications*; G.2.2 [**Discrete mathematics**]: Graph Theory – *Graph algorithms*.

Key words and phrases. clustering, complex networks, centrality measures, community detection, parallel computation.

more information can pass through. However, calculation of these values for each node is an extremely time-consuming procedure even though, in the past few years several more efficient algorithms have been developed. [6], [2], [14].

1.1. **Related work.** In this paper, we propose a novel cluster-based algorithm which addresses the issue of identifying influential nodes in complex networks using betweenness centrality. For clustering, we use the Louvain, Markov and Paris algorithm, these algorithms were introduced in [4] [18] [5]. Afterwards, we assign a cluster to each of the nodes of the network, we can calculate the centrality measure on these smaller sub-graphs thereby reducing the calculation time. We also use a modified version of this algorithm, where we introduce a parallel computation model.

The paper is organized as follows. Section 2.1 gives a brief overview of the definition of a graph, essential graph theory concepts and the definition of betweenness centrality. In this section, the examined clustering algorithms are also explained and in 2.3 the proposed algorithm was described. In Section 3, the implemented experiments are described in detail which is based on real-life networks to highlight the difference in performance between the proposed and the other methods. The conclusion and the possible future work about the issue in question can be seen in Section 4.

2. Overview of clusterization algorithms

2.1. **Preliminary and concepts.**

Definition 2.1 Adjacency matrix
An adjacency matrix is a $n \times n$ square matrix A, in which:

$$a_{ij} = \begin{cases} 1 & \text{if there exists an edge from } v_i \text{ to } v_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Definition 2.2 Betweenness centrality
Betweenness centrality is defined as

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

2.2. **Graph clustering.** The goal of graph clustering is to divide nodes into different clusters in a large graph based on specific criteria such as node connectivity or neighbourhood similarity. Clustering techniques are useful for the detection of densely connected groups in a large graph. In the following, we examine the following clustering algorithms: Markov, Louvain and Paris.

2.2.1. *Markov algorithm.*

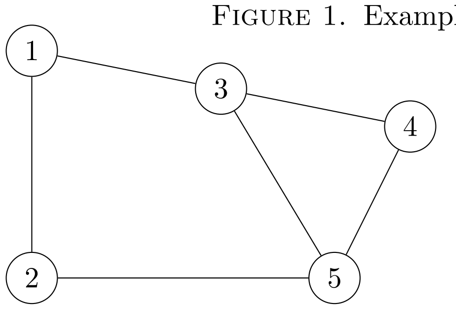
Definition 2.3 Markov matrix
A matrix A is a Markov matrix if its entries are greater than or equal to zero and each column's entries sum to one. Furthermore, each entry represents transition probabilities from one state to another.

Definition 2.4 Random walk
Let G be a graph or a digraph (directed graph) with the additional assumption that if G is a digraph, then $deg^{+}(v) > 0$ for every vertex v. Now let's have an object placed at vertex v_j. At each stage, the object must move to an adjacent vertex. The probability that it moves to vertex v_i is denoted as

(2)
$$m_{ij} = \begin{cases} \frac{1}{deg(v_j)} & \text{if } (v_j, v_i) \text{ is an edge in } G\text{, in case of digraph it is } \frac{1}{deg^{+}(v_j)} \\ 0 & \text{otherwise} \end{cases}$$

where m_{ij} represents the probability that a random walk of length k starting at vertex v_j, ends at vertex v_i, where the length is the number of edges.

FIGURE 1. Example of Markov matrix



$$M = \begin{bmatrix} 0 & 0.5 & 0.33 & 0 & 0 \\ 0.5 & 0 & 0 & 0 & 0.33 \\ 0.5 & 0 & 0 & 0.5 & 0.33 \\ 0 & 0 & 0.33 & 0 & 0.33 \\ 0 & 0.5 & 0.33 & 0.5 & 0 \end{bmatrix}$$

Random walk is a special case of the Markov chain, using the transition probability matrices. By walking randomly on a graph, we can find out where the flow tends to gather, and therefore where the clusters are located. This is the concept what Markov clustering and other clustering algorithms are based on.

Definition 2.5 Inflation
Given a matrix $M \in R^{k \times l}, M \geq 0$, and a real non-negative number r, the matrix resulting from re-scaling each of the columns of M with the power

coefficient r is called $\tau_r M$ and τ_r is called the inflation operator with power coefficient r.

$$\tau_r : \mathbb{R}^{k\times l} \longrightarrow \mathbb{R}^{k\times l}$$
$$(\tau_r M)_{pq} = (M_{pq})^r / \textstyle\sum_{i=1}^{k} (M_{iq})^r$$

In this case, the inflation operator is responsible for both strengthening and weakening of current while the parameter r controls the extent of this strengthening or weakening. Expansion is when we take the powers of the Markov Chain transition matrix, and it is responsible for allowing flow to connect different regions of the graph.

The algorithm converges to a "doubly idempotent" matrix which is at steady-state and there is only one value in a single column. To be able to identify clusters, the nodes are divided into two groups: attractors that attract other nodes, and nodes influenced by the previous group. Attractors always have at least one positive value within their row in the steady state-matrix and they draw those nodes that having a positive value in the same row. Nodes that have such a relationship can be classified in the same group.

FIGURE 2. Example of steady state matrix

$$\begin{bmatrix} 1 & - & 1 & - & - & - \\ - & - & - & - & - & - \\ - & 1 & - & 1 & - & - \\ - & - & - & - & 0.5 & 0.5 \end{bmatrix}$$

clusters: (1, 3), (2, 4), (5, 6)

The Markov clustering algorithm can be summarized as follows. Given an undirected graph, power parameter denoted as e, and inflation parameter denoted as r. First create the adjacency matrix, then add self loop to each node. After we have to normalize the matrix and expand it by taking the e^{th} power of it then inflate by taking inflation of the result with parameter r. Repeat steps until we reach the steady-state, finally we can obtain clusters.

2.2.2. *Louvain algorithm.*

Definition 2.6 Modularity
The modularity of a partition is a scalar value between -1/2 and 1 that measures the density of links inside communities as compared to links between communities. In case of weighted networks, it is defined by Newman and Girvan [15]

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where

- A_{ij} represents the edge weight between nodes i and j
- k_i and k_j are the sum of the weights of the edges attached to nodes i and j, respectively
- m is the sum of all of edge weights in the graph
- c_i and c_j are the communities of the nodes
- δ is Kronecker delta function ($\delta_{x,y} = 1$ if $x = y$, 0 otherwise)

Modularity can be used to compare the nature of the partitions collected by different methods. Louvain clustering can find high modularity partitions of large graphs and also unfold a complete hierarchical community structure for the network quite effectively.

To maximize modularity, the Louvain method uses two phases that are repeated iteratively. The algorithm's input is a weighted graph of n nodes. In the first step, we assign different community to each node of the graph which means we have n communities. After that, for each node i we calculate the gain of modularity what we can achieve by deleting i from its community and by putting into a community of j, where j is the neighbour of i. The gain in modularity ΔQ, when node i is moved into community C, can be calculated as:

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{2m} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

where

- $\sum_{in}$ is sum of the weights of the links inside C
- $\sum_{tot}$ is the sum of the weights of the links incident to nodes in C
- k_i is the sum of the weights of the links incident to node i
- $k_{i,in}$ is the sum of the weights of the links from i to nodes in C and m is the sum of the weights of all the links in the network

After this value is calculated for all communities i is connected to, i is placed into the community that resulted in the largest modularity increase, if there is no such increase then node i stays its original community. This process is applied to all nodes until no modularity increase can occur.

In the second step, it groups all nodes in the same community and creates a new network where the nodes are the groups of the previous step. Links between nodes of the same group are represented as self-loops while links between different communities defined as weighted edges between the community nodes.

2.2.3. *Paris algorithm.*

Definition 2.7 Weighted adjacency matrix
A weighted adjacency matrix is an A symmetric, non-negative matrix that for each pair of $(i, j) \in V, a_{ij} > 0$ iff there is an edge between node i and node j, and in this case a_{ij} is the weight of the edge $i, j \in E$.

Definition 2.8 Weight of a node
The weight of a node i is the sum of the weights of its incident edges

$$w_i = \sum_{j \in V} A_{ij}.$$

In the case of unit weights, w_i is the degree of node i. The cumulative weight of the nodes is

$$w = \sum_{i \in V} w_i = \sum_{i,j \in V} A_{ij}.$$

In the following, we will refer to C as a cluster which is a subset of V. The weights introduce a probability distribution on node pairs

$$\forall i, j \in V, \quad p(i, j) = \frac{A_{ij}}{w},$$

and also a probability distribution on nodes,

$$\forall i \in V, \quad p(i) = \sum_{j \in V} p(i, j) = \frac{w_i}{w}.$$

The distance between node i, j is defined as the node pair sampling ratio.

$$d(i, j) = \frac{p(i)p(j)}{p(i, j)}$$

The node distance can be also defined in a different way, which comes from the following conditional probability.

$$\forall i, j \in V, \quad p(i, j) = \frac{p(i, j)}{p(j)} = \frac{A_{ij}}{w_j}$$

Then the distance between i and j can be be written as

$$d(i,j) = \frac{p(i)}{p(i|j)} = \frac{p(j)}{p(j|i)}$$

Accordingly, consider a cluster C of the graph G. The weights induce a probability distribution on cluster pairs

$$\forall a, b \in C, \quad p(a,b) = \sum_{i \in a, j \in b} p(i,j)$$

and also a probability distribution on clusters,

$$\forall a \in C, \quad p(a) = \sum_{i \in a} p(i) = \sum_{b \in C} p(a,b)$$

and then the cluster pair sampling ratio is the distance between two different clusters a, b:

$$d(a,b) = \frac{p(a)p(b)}{p(a,b)}$$

Defines a conditional probability

$$\forall a, b \in C, \quad p(a|b) = \frac{p(a,b)}{p(b)}$$

which is the conditional probability of sampling a given that b is sampled, we have

$$d(a,b) = \frac{p(a)}{p(a|b)} = \frac{p(b)}{p(b|a)}$$

this distance will be used in Paris clustering algorithm to merge the closes clusters.

As in the Louvain algorithmn it starts with unique cluster therefore, every node is a cluster. In all phases, the two closest cluster will be merged. In section 2.2.2 we defined cluster modularity which can be rewritten in terms of probability distributions,

$$Q(C) = \sum_{i,j \in V} p(i,j) - p(i)p(j)\delta_C(i,j).$$

It can also be expressed from the probability distributions at the cluster level,

$$Q(C) = \sum_{a \in C} (p(a,a) - p(a)^2).$$

As Fortunato et al presented, maximizing the modularity has a resolution limit. Therefore Bonald and Charpentier [5] introduced a multiplicative factor γ, called the resolution. Then the modularity becomes:

$$Q_\gamma(C) = \sum_{i,j \in V} (p(i,j) - \gamma p(i)p(j))\delta_C(i,j)$$

Thus, the Paris algorithm can be understood as the deeply modified version of the Louvain algorithm, where the first phase is replaced by a simple merge.

2.3. **Proposed algorithm.** We now introduce our algorithm that can obtain betweenness centrality of large networks in a short time. We assume that out network G is undirected and consists of N nodes. The algorithm is divided into two-phase. First of all, we have to create clusters from the given network by using the introduced algorithms. Once the clusters are generated, we receive a mapping

$$\begin{aligned} c_1 &\longrightarrow [node\ ids] \\ &\cdots \\ c_n &\longrightarrow [node\ ids] \end{aligned} \tag{3}$$

where the keys are the cluster labels and the values are a list of nodes that belong to that cluster. These partitions form the basis of the method, therefore the mappings are saved in JSON format for later use. It is important to note that in the case of Markov clustering, the final result is greatly influenced by the value of the inflation parameter, therefore, the choice of this parameter requires a pre-calculation for each graph. To optimize this parameter, we selected numbers between $[1.1, 2, 6]$ intervals and then determined the value that produced the best modularity value.

In the second phase, we apply a concept similar to the divide and conquer programming strategy to graphs, as follows: first we have our large network G, then we load the previously generated cluster mapping and we create a list of sub-graphs based on the mapping. Thus, we obtain significantly smaller slices from the graph, on which the calculation of the centrality value is a much less expensive subproblem. Thereafter, we have to iterate through this list of sub-graphs defined by the clusters and execute the algorithm for calculating the betweenness centrality as described in [6].

To determine the betweenness centrality value for node v, we have to sum the pair-dependencies of all pairs on that node, where the pair-dependency was introduced as the dependency of a node $s \in V$ on a single node $v \in V$:

$$\delta_s(v) = \sum_{t \in V} \delta_{st}(v).$$

The calculation of the betweenness centrality can be split in two steps. First, we compute the length and the number of the shortest paths between all pairs then finally sum all pair-dependencies.

After the algorithm has performed on a single sub-graph, the sub-results are aggregated to obtain the centrality values for the whole graph.

Algorithm 1: Clusterized betweenness centrality

```
Input  : graph
Output: centrality-dict
centrality_dict ← {};
cluster_types ← ['Markov', 'Louvain', 'Paris'];
for type in cluster_types do
    cluster_mapping ← create_clusters(type);
    sub_graphs ← graph.subgraph(cluster_mapping);
    for sub_graph in sub − graphs do
        subresult ← calculate_bc();
        update(centrality_dict, subresult);
    end
end
return centrality_dict;
```

We also introduce a parallel version of the previous method which takes advantage of the multi-core computing capacity of today's modern processors. It contains the following changes: when we pass a sub-graph of G denoted as $G^{'}$ to calculate the betweenness centrality, the function that calculates centrality will accept a bunch of nodes and computes the contribution of those nodes to the centrality of the whole network.

Also, it divides the network into chunks of nodes, therefore those centrality measures can be calculated on separate different CPU cores as described in. [10] To determine the node of chunks, we take the particular sub-graph and divide it into $N/number\ of\ cpu\ cores$ pieces where N is the number of nodes in the sub-graph.

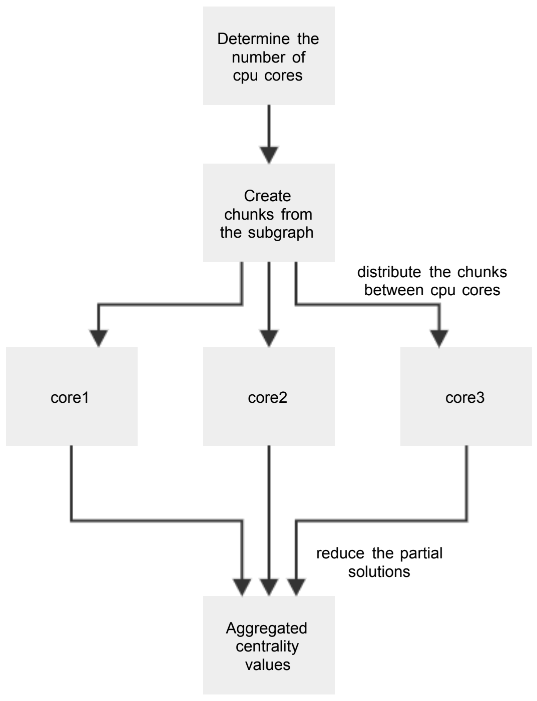

Determine the number of cpu cores
Create chunks from the subgraph
distribute the chunks between cpu cores
core1
core2
core3
reduce the partial solutions
Aggregated centrality values

FIGURE 3. Parallelized betweenness centrality

3. EXPERIMENTS

For our experiments, we used six actual networks to evaluate the performance of the examined method in case of big networks. The datasets are mostly from social sites, which served as an excellent basis for our observation, as they have many vertices and edges, and we also took into account road networks, web-networks. These networks are fb-pages-food, which consist of 620 nodes and 2100 edges, facebook-combined that has 4039 nodes and 88234 and power-bcspwr10 with 5300 nodes and 8300 edges. The fb-pages-politician includes 5900 nodes and 41700 edges. Also web-spam with 4800 nodes and 37400 edges, road-euroroad with 1200 nodes and 1400 edges. These networks can be gathered from [16] [12].

To demonstrate the efficiency of the proposed method, we made several measurements with different scenarios. As we mentioned earlier we chose betweenness centrality at the heart of our research therefore we will compare these calculated values with the values presented by the investigated method. In the proposed method we use Louvain, Markov and Parov clustering to calculate these centrality values. The experiments are divided into the following three parts and explained below.

From the clustering algorithms, only Markov clustering is not a parameter-free method, in which case the following inflation parameters were used: fb-pages-food (1.23), fb-pages-politician (1.17), while in the other networks inflation parameter was set to 2.

Table 1. Clusterization of the network

network	louvain	markov	paris
facebook-combined	16 (0.37)	10 (0.83)	6 (0.19)
fb-pages-food	19 ($-12 \cdot 10^{-4}$)	10 (0.65)	6 ($-6 \cdot 10^{-4}$)
fb-pages-politician	30 ($-25 \cdot 10^{-4}$)	23 (0.87)	8 ($-17 \cdot 10^{-4}$)
power-bcspwr10	50 (0.01)	30 (0.95)	4 (0.28)
road-euroroad	48 (0.41)	40 (0.88)	26 (0.09)
web-spam	2 ($-13 \cdot 10^{-4}$)	29 (0.50)	2 ($-4 \cdot 10^{-6}$)

Table 1 summarizes how the networks were split into clusters, the first numbers are the number of clusters while the values in brackets are the clusters' modularity. From this, it can be seen that the Louvain and Markov methods created order of magnitude a similar number of clusters while the Paris algorithm was able to detect quite a few groups within the same graph. In terms of modularity, it can be seen that the Louvain algorithm has performed best, while the other two algorithms have detected the lower density of links inside communities.

3.1. **Experiment 1.** In this experiment, we compared the centrality values for each node (referred to as simple) with values extracted from the proposed method.

Table 2. facebook-combined top 5 nodes

rank	simple	louvain	markov	paris
1.	107 (0.48)	437 (0.86)	0 (0.82)	0 (0.76)
2.	1684 (0.34)	0 (0.83)	107 (0.72)	107 (0.68)
3.	3437 (0.24)	3980 (0.80)	698 (0.32)	1684 (0.31)
4.	1912 (0.23)	1684 (0.66)	1085 (0.32)	1912 (0.29)
5.	1085 (0.15)	1912 (0.54)	862 (0.32)	1577 (0.22)

Table 3. fb-pages-food top 5 nodes

rank	simple	louvain	markov	paris
1.	265 (0.35)	483 (1.00)	265 (0.28)	265 (0.28)
2.	31 (0.16)	141 (0.90)	518 (0.16)	518 (0.16)
3.	518 (0.14)	96 (0.83)	31 (0.12)	31 (0.12)
4.	618 (0.09)	265 (0.80)	220 (0.09)	220 (0.09)
5.	35 (0.09)	49 (0.74)	217 (0.08)	340 (0.09)

Table 4. power-bcspwr10 top 5 nodes

rank	simple	louvain	markov	paris
1.	5268 (0.27)	5144 (0.58)	4774 (0.00)	5040 ($9.8 \cdot 10^{-6}$)
2.	3160 (0.26)	2645 (0.52)	2941 (0.00015)	4146 ($15 \cdot 10^{-5}$)
3.	5298 (0.26)	4074 (0.47)	1519 (0.00)	4898 ($14 \cdot 10^{-5}$)
4.	5040 (0.25)	4891 (0.47)	4671 (0.00)	5074 ($13.8 \cdot 10^{-5}$)
5.	4752 (0.23)	5232 (0.47)	5223 (0.00)	4870 ($13.5 \cdot 10^{-5}$)

Table 5. fb-pages-politician top 5 nodes

rank	simple	louvain	markov	paris
1.	5800 (0.27)	3008 (0.80)	5800 (0.16)	5800 (0.28)
2.	1864 (0.06)	5646 (0.74)	2900 (0.14)	3576 (0.07)
3.	3576 (0.05)	5699 (0.65)	4032 (0.09)	1864 (0.07)
4.	2900 (0.05)	5800 (0.64)	3094 (0.06)	1965 (0.06)
5.	1324 (0.05)	96 (0.56)	158 (0.05)	4032 (0.04)

Table 6. road-euroroad top 5 nodes

rank	simple	louvain	markov	paris
1.	401 (0.21)	58 (1.00)	111 (1.00)	111 (1.00)
2.	283 (0.21)	61 (1.00)	968 (0.17)	401 (0.20)
3.	276 (0.20)	646 (1.00)	1132 (0.17)	400 (0.20)
4.	452 (0.18)	18 (0.72)	1133 (0.17)	283 (0.18)
5.	451 (0.17)	917 (0.68)	1103 (0.13)	431 (0.17)

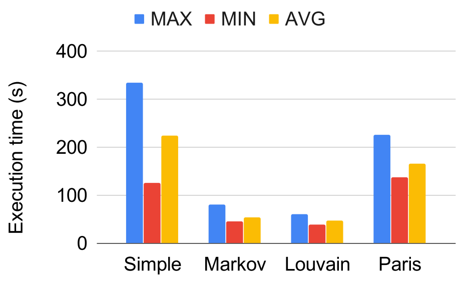
Table 7. web-spam top 5 nodes

rank	simple	louvain	markov	paris
1.	4044 (0.09)	3956 (1.00)	4044 (0.09)	4044 (0.09)
2.	981 (0.06)	327 (1.00)	981 (0.06)	981 (0.06)
3.	2561 (0.04)	709 (1.00)	2561 (0.05)	2561 (0.04)
4.	3114 (0.03)	1548 (1.00)	2916 (0.04)	3114 (0.03)
5.	3070 (0.03)	813 (0.95)	3070 (0.04)	3070 (0.03)

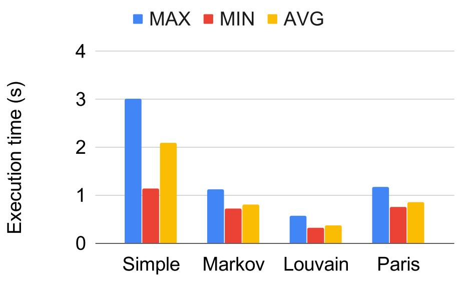
In the case of all networks, it can be seen that the investigated method mostly retained the nodes that were ranked among the top 5 peaks in the original calculation, only a few cases added new nodes to the top of the ranking.

3.2. **Experiment 2.** In this experiment, the efficiency and the performance were considered, resulting in a comparison of the runtime of the original betweenness centrality method and the proposed method.

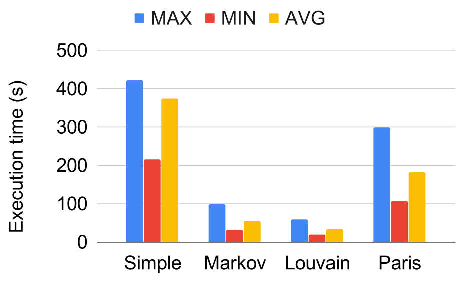
It can be seen that the investigated method's improved the performance of the calculation of betweenness centrality. In this aspect, the Markov method



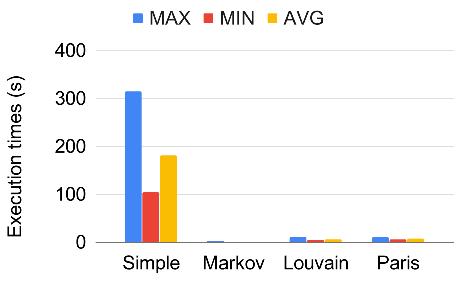
(A) facebook-combined



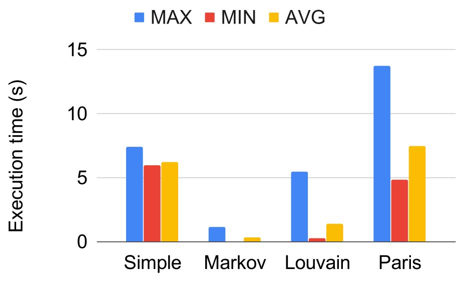
(B) fb-pages-food



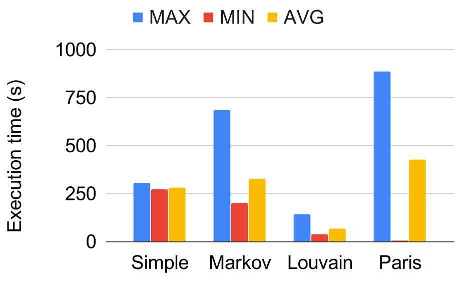
(C) fb-pages-politician



(D) power-bcspwr10



(E) road-euroroad

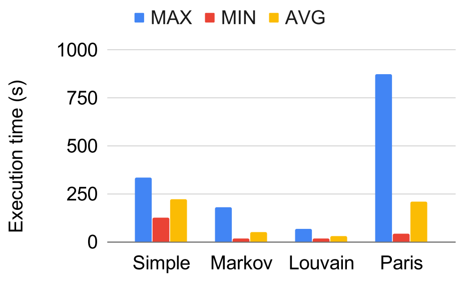


(F) web-spam

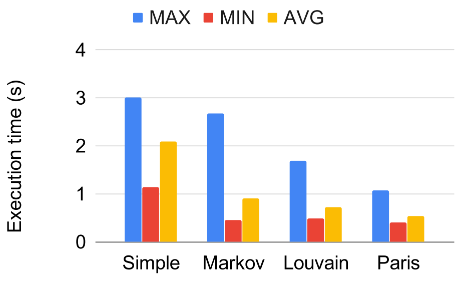
FIGURE 4. Performance with clustering

falls behind Louvain, while the Paris clustering outstandingly worse than the others. The reason behind this is, that clustering algorithm, as shown in the table above, has formed a fairly small number of groups, so it is not worthwhile to perform the division with such a low cluster number.

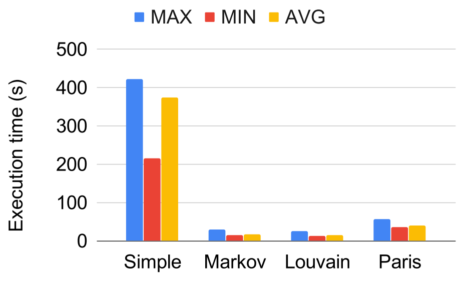
3.3. **Experiments 3.** In this experiment also the efficiency and the performance were considered as in the previous but in this case, we used parallel computation of betweenness centrality which was compared with the original betweenness centrality method.



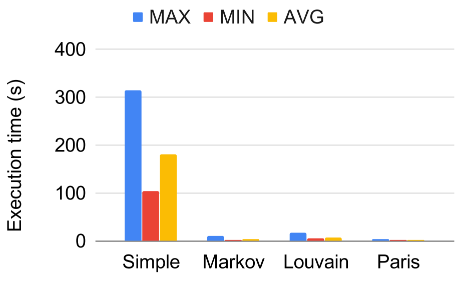
(A) facebook-combined



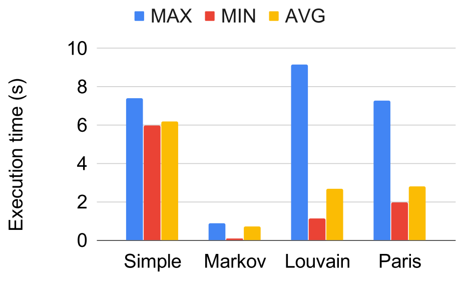
(B) fb-pages-food



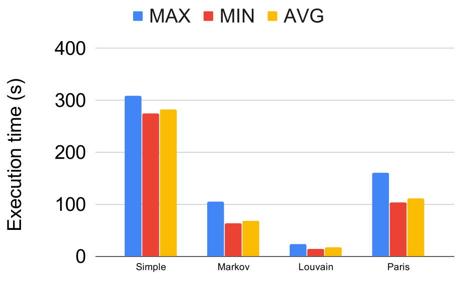
(C) fb-pages-politician



(D) power-bcspwr10



(E) road-euroroad



(F) web-spam

FIGURE 5. Performance with clustering and parallel computation

From these results, it can be seen that the computation time can be significantly reduced if we use clustering with parallel bc calculation. However, in some cases, spikes are visible in terms of maximum runtime. We have observed that if the graph fits in the memory the traditional calculation method is more profitable, and this method only worth for larger graphs.

4. Conclusion and future work

In this paper, we examined the performance of the proposed method to calculate betweenness centrality based on network clustering. This method calculates the centrality value for each node in the sub-graphs determined by different clustering methods. Our proposed solution is based on Louvain, Markov and Paris clustering algorithms. We investigated the method in large networks, to get a better picture of its performance. To determine the correctness of the method, we compared the values of the five most influential nodes obtained by the basic bc method with the values and nodes obtained by the method we proposed. The experimental results showed that the Louvain's performance was the best of all the investigated clustering methods since in most of the cases it was able to create an optimal number of clusters.

We only investigated the proposed method with betweenness centrality however, it is possible to use the same procedure with different centrality measures, which we intend to do in the future.

5. Acknowledgements

The project has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

References

[1] Aberer, K., Hauswirth, M., and Salehi, A. Infrastructure for data processing in large-scale interconnected sensor networks. In *2007 International Conference on Mobile Data Management* (2007), IEEE, pp. 198–205.

[2] Bader, D. A., Kintali, S., Madduri, K., and Mihail, M. Approximating betweenness centrality. In *International Workshop on Algorithms and Models for the Web-Graph* (2007), Springer, pp. 124–137.

[3] Bavelas, A. Communication patterns in task-oriented groups. *The journal of the acoustical society of America 22*, 6 (1950), 725–730.

[4] Blondel, V. D., Guillaume, J.-L., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment 2008*, 10 (2008), P10008.

[5] Bonald, T., Charpentier, B., Galland, A., and Hollocou, A. Hierarchical graph clustering using node pair sampling. *arXiv preprint arXiv:1806.01664* (2018).

[6] Brandes, U. A faster algorithm for betweenness centrality. *Journal of mathematical sociology 25*, 2 (2001), 163–177.

[7] Freeman, L. C. A set of measures of centrality based on betweenness. *Sociometry* (1977), 35–41.

[8] Gasteiger, J., and Zupan, J. Neural networks in chemistry. *Angewandte Chemie International Edition in English 32*, 4 (1993), 503–527.

[9] Gonzalez-Diaz, H., Vilar, S., Santana, L., and Uriarte, E. Medicinal chemistry and bioinformatics-current trends in drugs discovery with networks topological indices. *Current Topics in Medicinal Chemistry 7*, 10 (2007), 1015–1029.

[10] Hagberg, A., Swart, P., and S Chult, D. Exploring network structure, dynamics, and function using networkx. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[11] Irani, D., Balduzzi, M., Balzarotti, D., Kirda, E., and Pu, C. Reverse social engineering attacks in online social networks. In *International conference on detection of intrusions and malware, and vulnerability assessment* (2011), Springer, pp. 55–74.

[12] Leskovec, J., and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

[13] Mason, O., and Verwoerd, M. Graph theory and networks in biology. *IET systems biology 1*, 2 (2007), 89–119.

[14] Newman, M. E. A measure of betweenness centrality based on random walks. *Social networks 27*, 1 (2005), 39–54.

[15] Newman, M. E., and Girvan, M. Finding and evaluating community structure in networks. *Physical review E 69*, 2 (2004), 026113.

[16] Rossi, R. A., and Ahmed, N. K. The network data repository with interactive graph analytics and visualization. In *AAAI* (2015).

[17] Sen, S., and Wang, J. Analyzing peer-to-peer traffic across large networks. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment* (2002), pp. 137–150.

[18] van Dongen, S. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, 2000.

[19] Zaki, M. J., and Meira, W. *Data mining and analysis: fundamental concepts and algorithms.* Cambridge University Press, 2014.

Eötvös Loránd University, Budapest, Hungary
Email address: `n0qsdc@inf.elte.hu`

J. Selye University, Komárno, Slovakia
Email address: `kissae@ujs.sk`