A RED-BET Method to Improve the Information Diffusion on Social Networks

Son N. Duong[1], Hanh P. Du[2], Cuong N. Nguyen[3], Hoa N. Nguyen*[4]
Department of Professional Technology, VietNam Ministry of Public Security[1]
Department of Information Systems, VNU University of Engineering and Technology, Vietnam[2,4]
Department of Network Security, VietNam Ministry of Public Security[3]

***Abstract*—Information diffusion in the social network has been widely used in many fields today, from online marketing, e-government campaigns to predicting large social events. Some study focuses on how to discover a method to accelerate the parameter calculation for the information diffusion forecast in order to improve the efficiency of the information diffusion problem. The Betweenness Centrality is a significant indicator to identify the important people on social networks that should be aimed to maximize information diffusion. Thus, in this paper, we propose the RED-BET method to improve the information diffusion on social networks by a hybrid approach that allows to quickly determine the nodes having high Betweenness Centrality. Our main idea in the proposed method combines both the graph reduction and parallelization of the Betweenness Centrality calculation. Experimental results with the currently popular large datasets of SNAP and Animer have demonstrated that our proposed method improves the performance from 1.2 to 1.41 times compared to the TeexGraph toolkit, from 1.76 to 2.55 times than the NetworKit, and from 1.05 to 1.1 times in comparison with the bigGraph toolkit.**

***Keywords*—Information diffusion; graph reduction; betweenness centrality; parallel computing**

I. Introduction

Information diffusion is a key in social network analysis with many potential real-world applications. For example, it can be used for predicting large social events such as the Arab Spring [23], for improving the recommendation performance of products and services, and for maximizing advertising effect to individuals. However, it is difficult and time-consuming to be able to make the information diffusion forecast. The cause of this problem is the characteristic of the social network as a relatively large-scale graph with a large number of members (number of vertices) and number of relationships (number of edges) leading to the computation of the parameters for forecasting diffusion takes a lot of time [3].

Graph reduction, based on the application of graph theory [31], [36], is the basic and effective technique to minimize the time to calculate parameters in graphs and the time to analyze information diffusion forecast. Reducing unimportant vertices and edges on the graph will make the calculation faster. It is critical, however, to demonstrate that the reduction does not affect the graph's overall model.

Besides, in the problem of information diffusion, many parameters need to be calculated to conclude the diffusion. One of the most important parameters is centrality degree to identify the most critical (most central) vertice in a graph [4], which means the center of a pandemic, the main node on the Internet, or an influential person in social networks... [22] demonstrated the influence and the importance of centrality computation in the problem of disease diffusion as well as information diffusion.

Among the indicators of centrality degree [24], [28], besides the Closeness Centrality, the Eigenvector Centrality, the Betweenness Centrality (BC) [12] is an important measure, valuable in determining a vertice is an intermediate bridge when establishing the shortest relationships (path) between other vertices. This centrality has been used in many practical problems. Such as, during the current Covid-19 pandemic, the authors in [32] used the BC indicator to identify subjects that need to be localized soon to proactively prevent the spread of SARS-CoV-2 coronavirus. In [13], [19], authors proved that BC has an important influence on the acceptance and diffusion of information. But, to compute BC for all vertices in a graph, we must solve the problem of finding the shortest path on all pairs of vertices in the graph, a process that takes a long time with large graphs.

To accelerate the computation, a number of researches have been actively performed, focus on the orientations such as graph reduction [1], [14], [33], approximating [6], [17], parallelizing using GPU [7], [15], [25] or using a high-performance computing platform [2], [11]. Based on some previous studies, in this paper, we propose a hybrid method to improve the information diffusion on social networks by a hybrid approach that allows us to quickly determine the nodes having high BC. Our method combines two processes: graph reduction and BC parallel computing to accelerate the computation, analyze information diffusion forecast while proving that our method did not affect the overall model of the graph.

The remainder of this paper is divided into four sections as follows: Section II describes fundamental concepts and definitions; other related works will be also presented in this section. Our hybrid method to improve information diffusion by graph compression and quickly determining the BC on social networks will be given in Section III. Section IV is dedicated to present the experiment results and evaluations confronted with the bigGragh and NetworKit, TeexGraph toolkits. Finally, the conclusion and future works will be presented in Section V.

II. Preliminaries and Related Work

A. Notation

Social networks are represented in the form of graphs. In which, the graph is a flexible data structure, represented as a

set of vertices and edges, otherwise known as a set of nodes and their relationships [26], [27].

Definition 1.1. In graph theory, a graph G, denoted $G = (V, E)$, is made up of a set of nodes V and a set of edges E connecting nodes with $E = \{(v_i, v_j)|v_i, v_j \in V\}$. This structure allows us to model all kinds of problems, from building disease transmission maps, mapping systems, and graphing of the relationships on Social Networks,...

For the convenience of evaluation, in this paper, we only focus on unweighted graphs, but can be directed or undirected. It means:

- Unweighted Graph: is a graph with no value assigned to each edge.

- Simple Graph: is an unweighted graph and only has exactly one edge between two vertices.

- Connected Graph: is a graph of existence path (u, v) from vertice u to v for all pairs of vertices $u, v \in V$.

In this paper, we also use some concepts such as:

- Degree of the vertice, denoted $deg(v)$ is the total number of edges to and from vertice v. Due to considering only graphs, the degree of the vertice v is also the number of vertices adjacent to that vertice. The vertice v is called the hanging vertice if $deg(v) = 1$ and is called the isolated vertice if $deg(v) = 0$. In a directed graph, the degree of the vertice is further divided into d_{out} and d_{in}.

- Path is a series of vertices and edges from vertex u to vertex v in the graph without any vertices or edges repeated.

- The shortest path between two vertices u, v, denoted $dist(u, v)$, with an unweighted graph, the shortest path is the path with the smallest number of edges from u to v. If u, v overlap $dist(u, v) = 0$, if u, v don't link dist $(u, v) = \infty$.

- Adjacent list $\Gamma(v)$ is a list of vertices adjacent to v, also known as its neighbor set of v.

Definition 1.2. The Degree Centrality of vertice v is the number of edges associated with vertice v. This measure is determined by the following formula:

$$CD(v) = deg(v) : v \in V \tag{1}$$

Thus, the Degree Centrality of a vertice is the degree of that vertice. With the directed graph, this measure is also divided into CD_{out} and CD_{in}. For social networks, the Degree centrality is a person's number of friends on Facebook or the number of followers on Twitter.

Definition 1.3. The Betweenness Centrality of a vertice v is calculated by the following formula [29]:

$$BC(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2}$$

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

Thus, BC is the number of intermediaries one person takes on when establishing the shortest relationships among others [12]. In his study, Linton Freeman conceived that the vertices with high probability lying on the shortest path between two randomly selected vertices in the vertice set V would have the most BC.

B. Related Work

In the studies of accelerating computation and analysis, graph reduction is considered a highly effective method. The essence of graph reduction is to remove/replace unnecessary/unimportant vertices and edges to obtain a more compact graph and retain important vertices and necessary properties of the graph.

Feder and Motwani in [33], Adler and Mitzenmacher in [14] mentioned reduction by graph compression. In it, [33] offers a compression method by using a partitioning algorithm for the bipartite graph. The author in [14] put the compression problem on the problem of finding the minimum spanning tree. Basically, the graph compression method results in a more compact graph, however, it often serves the problem of graph storage (data or structure). For the social network analysis problem with constantly changing data, the graph compression is not suitable because continuous conversion between the original graph and the compressed graph is not positive.

Gilbert and Levchenko in [1] gave several graph reduction methods with two algorithms KeepOne, KeepAll, and the method of deleting redundant vertices RVE (Redundant Vertex Elimination). In it, the KeepOne algorithm is similar to the method of Adler and Mitzenmacher, that is to find a minimum spanning tree for the graph. This method allows to keep the maximum number of important vertices and the vertices located between those important vertices, however, the biggest disadvantage of this method is that it does not preserve the shortest path between the two vertices. In contrast to it, KeepAll is an algorithm that allows retaining the shortest path between important vertices. It can be said that the two algorithms have their own strengths and are suitable in certain cases (for example, KeepOne will be suitable for the network planning problem, KeepAll is suitable for the problem of finding the shortest way in traffic), however, they are not suitable for social network analysis problems. RVE is the closest method to analysis and computation on social networks, this method allows the elimination of vertices that have a common adjacent one. The method is often applied in communication network reduction with the elimination of unnecessary redundant nodes. If applied in social network analysis, it can delete the unimportant vertices. But besides that, if the important vertices have a common adjacent one, one of which is also removed.

In previous research, we relied on the idea of the redundant vertex elimination - RVE method, which is to reduce the graph based on the replacement of the equivalent 1-degree vertice. That is, in the reduction process, we only consider the hanging vertices, $deg(v) = 1$ equivalent to $BC = 0$. In that research, we have given the solution, described the algorithm, and experimented on the small simulated network of 100 vertices. In this paper, in addition to the parallelization, we also came up with the specific pseudo-code algorithm, proved

the effectiveness of the method in the problem of information diffusion, and experimented with it on some larger datasets.

For the quick BC computation, this is also the content that many researchers are interested in. At first, there are several methods to solve APSP problems, such as Floyd-Warshall's algorithm, Johnson's algorithm [34] and Brandes algorithm [35]. Compared with calculating APSP by Floyd-Warshall algorithm (computation complexity $O(|V|^3)$)) and Johnson's algorithm (complexity $O(|V|^2 log(|V|) + |V||E|)$)), then Brandes' algorithm with time complexity is $O(|V||E|)$ on an unweighted graph and $O(|V||E| + |V|^2 log(|E|))$ on a weighted graph is considered the most effective algorithm to compute BC for all vertices in graph.

Riondato and Kornaropoulos in [17], Mahmoody, Tsourakakis and Upfal in [6] give an idea of the fastness approximation computing the BC based on sampling technique. According to this method, some of the shortest paths will be randomly sampled, thereby applying the algorithm to estimate the distance between the vertices and approximate the BC.

Bernaschi, Carbone and Vella in [15], Fan, Xu and Zhao in [25], McLaughlin and Bader in [7] gave advanced solutions to speed up BC computation by parallelizing computation processes using GPU graphics processors. The author in [15] also combines multiple GPU sets and parallel models using distributed memory MPI to speed up computation. Notably, these methods all use the classic Brandes algorithm.

Ching in [2], Wei, Chen, Zhou, Zhou, and He in [11] suggest using the GraphLab and Apache Giraph toolkits on complex computation infrastructures such as cluster computer systems or high-performance supercomputers. However, these toolkits are mainly designed to analyze very large networks up to trillions of edges, not really efficient for computation with real networks not too large as Facebook.

In addition to the above methods, improving BC computation performance can be applied by parallelizing the NetworKit toolkit or the TeexGraph toolkit for large-scale social network analysis. These tools all use a parallel shared memory model and use the OpenMP library to parallelize BC metrics. One of our research in [21] is the parallelization of SSSP calculations in Brandes algorithm with parallel thread programming model on CPU and using CilkPlus library.

In the above studies, there is currently no research focusing on combining the process of graph reduction and improving the performance of calculating the betwenness centrality when solving the problem of information dissemination on social networks. This is the main motivation for us to propose a method overcoming this challenge in this paper.

III. Red-Bet Hybrid Method to Accelerate the Computation in Information Diffusion Problem

Based on the above analysis, in this paper, we propose a hybrid method to improve the information diffusion on social networks represented by an unweighted graph. This method is based on the combination of two processes:

i. The graph reduction is based on replacing the equivalent 1-degree vertices.

ii. Paralleling SSSP calculations in Brandes algorithm with parallel thread programming model on CPU using CilkPlus library.

Since then, we call the proposed hybrid method namely **RED-BET** .

In this study, we still use a graph data structure approach like [20] to improve the cache hit rate when referring to graph data. Thus, the large-scale graph $G = (V, E)$ is organized by the adjacent vertex lists where each vertex is assigned an identifier from 0 to $|V|-1$. For edge data, sorted vertex vectors are used to represent the graph edges. That means the graph edges are structured in the vectors array $Edges[u]\ \forall u \in V$. For real-world social networks, such as Twitter or Facebook, the vertices number is small than 2^{32}. Therefore, the graph data is allocated by a 4-byte integer vectors array and our method can analyze a graph with the largest number of vertices possible being 2^{32}.

A. Graph Reduction

The first step in the reduction of the graph is to determine the equivalent 1-degree vertices. 1-degree means hanging vertices $deg(v) = 1$ and equivalent mean that they must have a duplicate set of adjacent vertices $\Gamma(v)$. And since they are 1-degree vertices (with a single adjacent vertice), we can say that we need to find the hanging vertices that have a common adjacent vertex.

To do this, we need to proceed with graph searching [30], [18], [9]. While considering whether to use breadth-first searching (BFS) or depth-first searching (DFS), we found that the first phase of Brandes' BC computation that we improved was the browse by BFS. Therefore, we proceed to integrate the graph reduction into the graph searching phase, ie also using the BFS method.

After determining the equivalent vertices above, replacing them is understood that we will choose a single vertice to represent, or that is, delete the equivalent vertices and leave a single vertice. This reduction reduces the size of the graph, which in turn will certainly change the result of the BC computation of vertices. However, there are two reasons for us to decide to still reduce the graph. Firstly, the vertices that we replace are the hanging 1-degree vertices, the Betweenness Centrality $BC = 0$, meaning the vertices are not important, their removal does not affect the graph too much.

Secondly, in analyzing the social network graph, we care about which vertice is "most important", or which vertice has the highest centrality or the highest BC not about what is the exact BC of that vertice. Our reduction method allows to ensures that the "most important" property of those vertices is preserved.

Thus, the graph reduction algorithm is illustrated as follows:

Algorithm 1: Graph Reduction Algorithm

Input: $G = (V, E)$, is organized as an vector array $Edges[][]$
Data: queue $Q \leftarrow$, stack S create the hollow (empty) and its able to contain $|V|$ vertices
Output: $Edges[][]$ of G have been reduced

```
/* Phase 1. Graph reducing          */
```
foreach $v \in V$ **do**
  while Q *not empty* **do**
    $v \leftarrow Q.pop(); S.push(v)$;
    foreach
     $w \in Edges[v] \&\& Edges[w].Size() = 1$ **do**
      if $u \in Edges[v] \&\& Edges[u].Size() = 1$
       then $Edges[v] \leftarrow Edges[v] \backslash \{u\}$;
       /* delete u from Adjacent list of v */
       $Edges[u] = \{\}$; /* delete vertice u */
       ;
    end
  end
end
return $Edges[][]$;

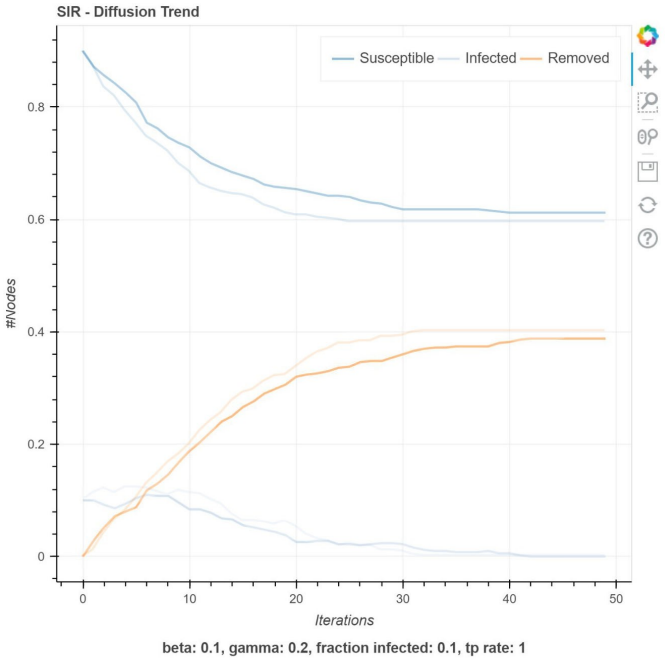

Fig. 1. Test with Random Graph of 500 Vertices.

To prove the effectiveness of graph reduction in the Information diffusion problem, based on the method and library (NDlib) in [8], we tested the information diffusion process on some random graphs using the SIR model and compare it with that process when the graph is reduced. For the first time, we used a random graph of 500 vertices, the probability to generate edges is 0.005, with 50 times the information was diffused. The results are shown in Fig. 1 with the light line describing the ratio of the graph before reduction and a bold line describing the ratio of the graph after reduction. The results showed that the ratios of Susceptible, Infected, and Removed had not much difference between the two graphs.

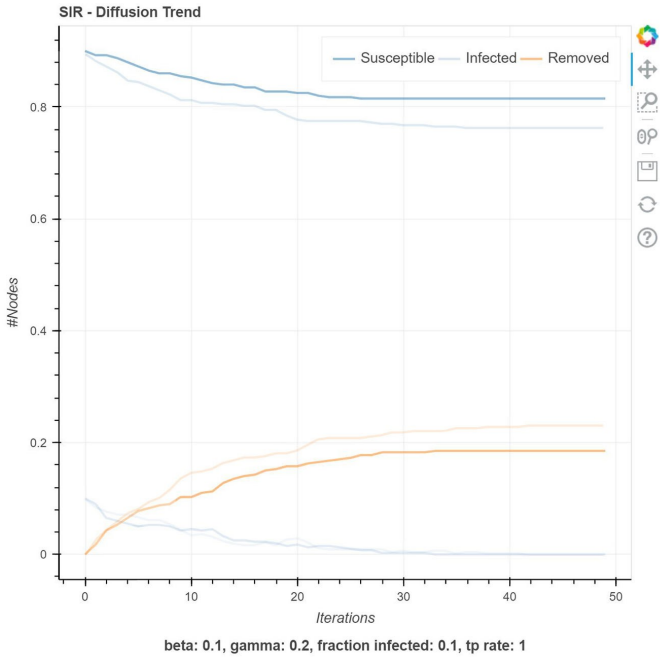

Fig. 2. Test with Random Graph of 400 Vertices.

TABLE I. COMPARE THE DIFFUSION TIME (SECONDS)

Times	Graph with 500 nodes		Graph with 400 nodes	
	Before	After	Before	After
1	1.30	1.25	1.28	1.20
2	1.27	1.23	1.26	1.20
3	1.28	1.25	1.25	1.19
4	1.32	1.24	1.26	1.20
5	1.28	1.22	1.25	1.18
6	1.28	1.24	1.28	1.21
7	1.29	1.23	1.24	1.18
8	1.32	1.25	1.24	1.19
9	1.31	1.25	1.25	1.18
10	1.30	1.24	1.26	1.20

The second time, we used a random graph of 400 vertices, the probability to generate edges is 0.004, with 50 times the information was diffused. The results in Fig. 2 show the same result. The total time after performing 10 times for each experiment is shown in Table I.

The results from Fig. 1, 2, and Table I showed the effectiveness of the graph reduction method in speeding up the process of analyzing and diffusing information.

B. Paralleling the Betweenness Centrality Computation Process

First of all, to represent graphs there are three main methods: (1) edge list, (2) adjacent matrix, and (3) adjacent list. In a relatively large-scale graph, the edge-list method is quite simple, but the calculation on the graph such as inserting, deleting vertices is difficult. The adjacent matrix method is also not usable due to memory size limitations. Therefore, the most suitable is the adjacent list method. Thus, for vertice data, in the graph $G = (V, E)$, each vertice is assigned a value from 0 to $|V| - 1$. For edge data, vertice vectors are arranged to represent the edges of the graph, or edge data will be represented in the vector array.

Secondly, the BC computation by the method of Brandes depends largely on the BFS graph searching process. To reduce the size of the queue when searching, each time we search a vertex u, we will use the $Maps$ array where the v-th bit position represents the searching state or not of vertice v. The $Queue$ queue is also organized to store the shortest distance from u to the searched-vertice in the $Queue$. Due to the large size of the $Queue$ queue and searching bookmark $Maps$ array (with the number of elements equal to V), the memory allocation will take a lot of time. So we will preallocate the memory containing these arrays corresponding to the number of threads that can execute in parallel.

Thirdly, to be able to exploit the performance of multi-core CPUs, our plan to parallelize BC computation is to execute BC computations in parallel on different vertices, not parallelize the searching process and compute the shortest path from one vertice to all other vertices (SSSP). This approach allows SSSP searching to be performed in specialized threads, thereby improving cache access speed.

Fourthly, the libraries for parallelization, such as CilkPlus, OpenMP, and Pthread, Leist and Gilman in [5] experimented and demonstrated the CilkPlus library give a better speed-up factor than OpenMP and Pthread. Accordingly, we will use the Cilkplus library to install the parallel computation.

Finally, it can be said that computing on the graph with a large number of vertices and edges, in which the BC is a relatively difficult problem in parallel. The reason is that in Phase 2 of the algorithm, the accumulation process requires a simultaneous control technique to process the accumulated data from parallel threads. During research and testing, we added a $reducerBC[v]$ vector in the Cilkplus library [16]. Technically, $reducer$ allows creating a separate cumulative variable for each thread, and combining its own cumulative variable will result in the correct order when the threads end. That is $reducerBC[v]$ vector allows to update the BC value of vertice v when executed in parallel with the Cilkplus library.

From there, the algorithm to compute BC in parallel has been given in [21] and shown in Algorithm 2.

As can be seen, the time complexity of algorithm 1 is $O(|V|*|E|)$ and algorithm 2 is $O(\frac{(|V|*|E|)}{t})$. So with algorithm 2, if deployed algorithm with a thread $t = 1$, it would equal the complexity of Brandes' base algorithm of $O(|V| * |E|)$. However, if this algorithm is executed in parallel with t threads, the time complexity of the algorithm will be reduced by t times.

Algorithm 2: Computation of BC in parallel

Input: $G = (V, E)$ have been reduced, is organized as $Edges[][]$ vector field
Data: queue $Q \leftarrow$, stack S create the hollow (empty) and its able to contain $|V|$ vertices ;
$dist[v]$: to save the distance from the source vertice to v ;
$Pred[v]$: to store the list all the vertices on the shortest path from the source vertice to v ;
$\sigma[v]$: the number of shortest paths from the source vertice to v ;
$\delta[v]$: the number of shortest paths from the source vertice through v ;
$reducerBC[v]$: vector contains BC values of all vertices v and allows the concurrency update in parallel with CilkPlus library;
Output: $BC[.]$ for any $v \in V$

```
/* Execute in parallel using CilkPlus
   library                                  */
```
for $s = 0$ **to** $Edges.size()$ **do**
```
    /* Phase 2. Graph researching    */
```
    foreach $v \in V$ **do** $Pred[v] \leftarrow$ empty list;
    $dist[v] \leftarrow \infty; \sigma[v] \leftarrow 0$;
    $dist[s] \leftarrow 0; \sigma[s] \leftarrow 1; Q.push(s)$;
    while Q *not empty* **do**
        $v \leftarrow Q.pop(); S.push(v)$;
        foreach $w \in Edges[v]$ **do**
            if $dist[w] == \infty$ **then**
                $dist[w] \leftarrow dist[v] + 1; Q.push(w)$;
            if $dist[w] == dist[v] + 1$ **then** $\sigma[w] \leftarrow \sigma[w] + \sigma[v]; Pred[w].push\_back(v)$;
        end
    end
```
    /* Phase 3. Accumulation         */
```
    foreach $v \in V$ **do** $\delta[v] \leftarrow 0$;
    while S *not empty* **do**
        $w \leftarrow S.pop()$;
        for $v \in Pred[w]$ **do**
            $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]}.(1 + \delta[w])$;
        if $w \neq s$ **then**
            $reducerBC[w] \leftarrow reducerBC[w] + \delta[w]$;
    end
end
$reducerBC.move\_out(BC)$;
```
/* to return the results to vector BC */
```
return $BC[.]$;

C. Red-Bet Hybrid Method

The proposal RED-BET hybrid method is based on both the graph reduction and paralleling the BC computation process. First of all, the graph reduction is performed before the BFS phase in order to reduce both the vertices and the edges of graph. After that the BC computation process is executed to find the most important vertices in the social network. However, to consolidate the information diffusion, we have to prove the two mentioned contents above that the vertices that we replace are not important, their removal does not affect the graph too much and the "most important" property of those vertices is preserved.

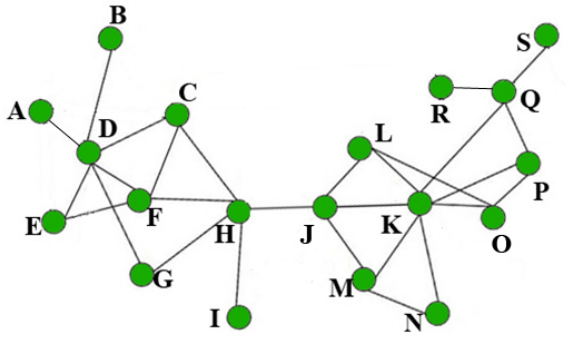

Fig. 3. Graph Before Reducing.

TABLE II. RESULT OF COMPUTING BC AND CC BEFORE REDUCING

Vertice	BC	CC	Vertice	BC	CC	Vertice	BC	CC
A	0	0.24	H	176	0.44	O	1	0.31
B	0	0.24	I	0	0.31	P	3	0.32
C	24	0.36	J	163	0.45	Q	66	0.33
D	71	0.31	K	133	0.42	R	0	0.25
E	0	0.29	L	10	0.37	S	0	0.25
F	49	0.37	M	10	0.37			
G	24	0.35	N	0	0.31			

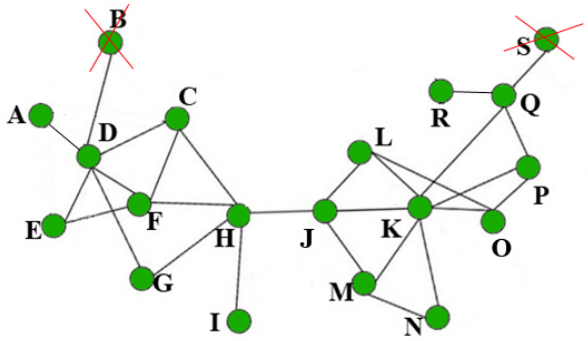

Fig. 4. Graph After Reducing.

To prove that we experiment on a simple graph with 19 vertices, 28 edges as shown in Fig. 3. Applying the formula to calculate the BC and the Closeness Centrality (CC), we get the results as Table II.

Thus, H is the vertice with the highest Betweenness Centrality $BC[H] = 176$ (then to the vertices J and K), J is the vertice with the highest Closeness Centrality $CC[J] = 0.45$ (after to the vertices H and K). With a simple small graph, intuitively we can see that vertices A and B are 2 equivalent 1-degree vertices, similarly vertices R and S are 2 equivalent 1-degree vertices. Apply the proposed graph reduction method, we obtain the graph after reduction as in Fig. 4.

Applying the formula for calculating the BC and CC with the graph after reduced, we get the results as Table III. After reduction, H is still the vertice with the highest Betweenness Centrality $BC[H] = 140$ (then to the vertices J and K), J is also the vertice with the highest Closeness Centrality $CC[J] = 0.48$ (then to the vertices H and K). Thus, it can be concluded that graph reduction by the proposed method, although changing the centrality value, does not affect the properties and important vertices of the graph for the diffusion information analysis problem.

TABLE III. RESULT OF COMPUTING BC AND CC AFTER REDUCING

Vertice	BC	CC	Vertice	BC	CC	Vertice	BC	CC
A	0	0.24	H	140	0.47	O	1	0.33
B	-	-	I	0	0.33	P	2	0.33
C	14.67	0.37	J	129	0.48	Q	30	0.33
D	35	0.31	K	98	0.44	R	0	0.25
E	0	0.30	L	9	0.39	S	-	-
F	37.67	0.38	M	9	0.39			
G	14.67	0.36	N	0	0.32			

For validating this method, we will conduct experiments in the next section to prove that: (1) the reduction brings more efficiency in parallelization than the non-reduction, that is, the experimental results of this research's algorithm are more optimal than the bigGraph algorithm [21]; (2) the speed-up factor of the proposed algorithm compared to the two tools NetworKit and TeexGraph is clearer than the speed-up factor of bigGraph algorithm when compared to the two tools NetworKit and TeexGraph.

IV. EXPERIMENT AND EVALUATION

To evaluate the effectiveness of the hybrid method in accelerating the BC calculation, we installed our algorithm on the computer with 2 x CPU (2-cores per CPU) configuration, 128 GB main memory, operating system CentOS Linux release 7.4.1708, gcc 7.2.0 compiler. This computing system was configured with a maximum of 36-threads in parallel.

A. Datasets

To test the above algorithm, we have collected social network data sets published from two major organizations, SNAP [10] and Aminer Datasets for Social Network Analysis [37] including:

- *ego-Facebook*: A data set built from Facebook's friends' lists. These lists are gathered from members participating in the Facebook application-based survey (DS1).

- *gemsec-Facebook*: A data set consisting of 8 subnets built to represent legitimate Facebook pages. These Facebook pages are modelled with vertices and edges representing the links between those pages. Due to the size and time limitations, we only selected two large networks in the gemsec-Facebook dataset for testing: Politician (DS2) and Artist (DS3).

- *com-DBLP*: This is a dataset that represents the DBLP co-authorship network (DS4).

- *com-Youtube*: This dataset is collected from the ground-truth communities in Youtube social network (DS5).

The above data sets are all connected graphs, their main characteristics are illustrated in Table IV.

TABLE IV. EXPERIMENT DATASET CHARACTERISTICS

Dataset	Edges	Nodes	Diameter
ego-Facebook (DS1)	88,234	4,039	8
gemsec-Facebook Politician (DS2)	41,729	5,908	14
gemsec-Facebook Artist (DS3)	819,306	50,515	11
DBLP (DS4)	1,049,866	425,957	23
Youtube (DS5)	2,987,624	1,157,828	24

TABLE V. EXECUTION TIME (SECONDS) TO COMPUTE BC

Threads	DS1		DS2		DS3	
	bigGraph	RED-BET	bigGraph	RED-BET	bigGraph	RED-BET
1	3.03	2.85	8.20	7.74	1129.47	1065.53
4	1.51	1.42	4.52	4.25	556.46	524.98
16	0.54	0.51	1.60	1.51	196.46	185.34
36	0.23	0.22	0.74	0.70	99.85	94.201

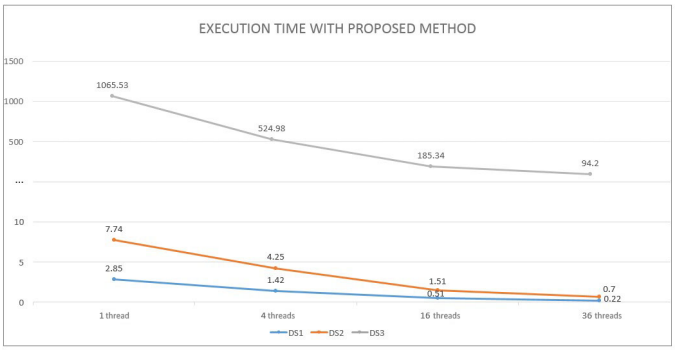

Fig. 5. Execution Time (Seconds) to Calculate BC of RED-BET.

B. Results and Evaluation

It should be noted, our previous research results in [21] have proven that executing more parallel threads will increase the speed up (i.e. time to compute BC less). Therefore, in this study, we do not re-prove the above content and only focus on proving two contents as mentioned in Part II, that is: (1) the reduction method brings more parallelization effect than the non-reduction, it means that the experimental results of this study (RED-BET) are more optimal than the bigGraph algorithm in the study [21]; (2) the speed-up factor of RED-BET compared to the two tools NetworKit and TeexGraph is clearer than the speed-up factor of bigGraph algorithm when compared to the two tools NetworKit and TeexGraph.

First, we tested the RED-BET algorithm and bigGraph algorithm with three datasets DS1, DS2, DS3 with 4 circumstances 1, 4, 16, 36 parallel threads. The aggregated results were based on the average execution time after 10 test runs for each solution and presented in Table V.

The result above shows that the proposed algorithm RED-BET combines graph reduction and parallelization have a smaller execution time, ie better performance than the bigGraph algorithm in previous studies [21]. To visualize, Fig. 5 shows the change of BC computation time with RED-BET algorithm when the number of parallel threads changes similar to the bigGraph algorithm.

Obviously, combined with the research [21], the proposed algorithm RED-BET would also be more efficient than the NetworKit and TeexGraph toolkits. However, to be objective in research, we continue to test the RED-BET algorithm with two datasets DS4, DS5 to compare and evaluate the speed-up factor compared to NetworKit and TeexGraph. In all tests, we used 36 parallel threads (the maximum number of threads on the system). Table VI illustrates the average time of 10 executions of the BC center calculation of the four solutions in the test.

TABLE VI. TIME TO COMPUTE BC (SECONDS)

Datasets	RED-BET	bigGraph	TeexGraph	NetworKit
DS1	0.22	0.23	0.31	0.56
DS2	0.70	0.74	0.84	1.70
DS3	94.20	99.85	110.58	234.12
DS4	2193.54	2345.62	2694.78	4823.47
DS5	50977.15	56071.60	68744.80	90522.30

TABLE VII. SPEED UP FACTOR OF RED-BET COMPARED WITH BIGGRAPH, TEEXGRAPH AND NETWORKIT WHEN COMPUTING BC

Datasets	RED-BET / bigGraph	RED-BET / TeexGraph	RED-BET / NetworKit
DS1	1.05	1.41	2.55
DS2	1.06	1.2	2.43
DS3	1.06	1.17	2.49
DS4	1.07	1.23	2.2
DS5	1.15	1.35	1.76

Detailed representation of the time to compute the BC of four solutions is illustrated in Fig. 4. Table V and Fig. 6 confirmed that the RED-BET algorithm has a smaller execution time than the TeexGraph and NetworKit toolkits. The speed up factor of the proposed algorithm compared to the bigGraph algorithm and the two tools TeexGraph and NetworKit are shown in Table VI.

Thus, for all three datasets, the RED-BET algorithm gives better performance, computing the BC in the shortest time compared to the tools TeexGraph, NetworKit as well as bigGraph algorithm in the previous study. Table VII also shows the speed-up factor of RED-BET is clearer when compared to TeexGraph, NetworKit, 1.2 to 1.41 times when compared to TeexGraph and 1.76 to 2.55 times when compared to NetworKit.

V. CONCLUSION

In this paper, we focus on accelerating the computation in information diffusion on the social network. The proposed RED-BET method is based on both the graph reduction and the parallelization of BC computation. Specifically, it combines the process of reducing the graph based on replacing the equivalent 1-degree vertices and parallelizing the SSSP calculations in Brandes' algorithm with the parallel thread programming model on the CPU and using the CilkPlus library. The time complexity of the proposed algorithm is $O(\frac{(|V|*|E|)}{t})$, where t is the number of parallel threads.

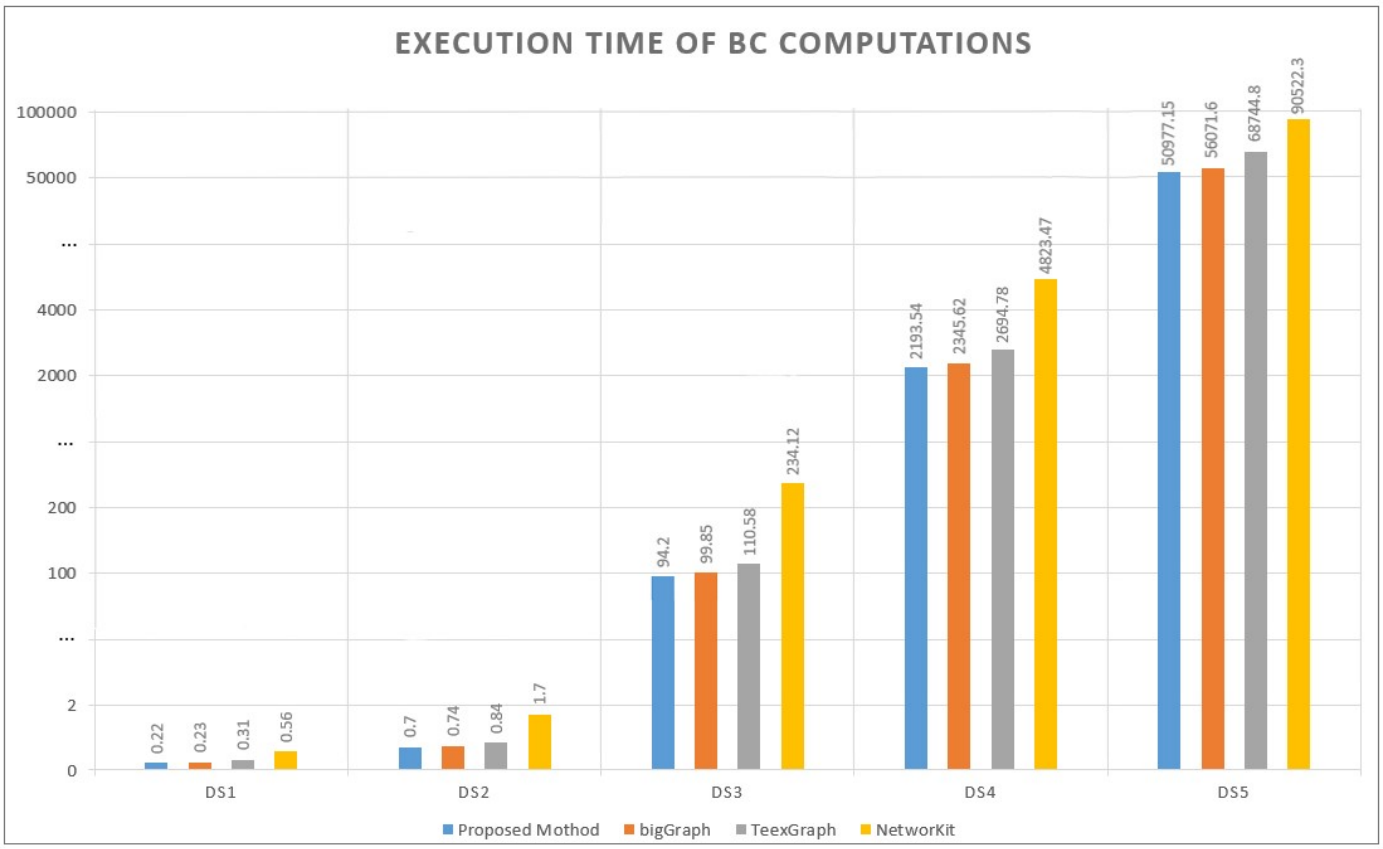

Fig. 6. Evaluation of BC Execution Time for Different Solutions (Seconds).

Our RED-BET algorithm was tested with some datasets published from two major organizations, SNAP and Aminer. Test results show that our solution is more efficient 1.2 to 1.41 times than the TeexGraph toolkit, 1.76 to 2.55 times than the NetworKit toolkit, and 1.05 to 1.1 times than the bigGraph.

In future work, we will focus on expanding our method to accelerate the calculation of other metrics in information diffusion analysis on the social network.

REFERENCES

[1] A. C. Gilbert, K. Levchenko, "Compressing Network Graphs", Proceedings of the LinkKDD Workshop at the 10th ACM Conference on KDD, 2004.

[2] A. Ching, "Scaling apache giraph to a trillion edges", https://www.facebook.com/notes/facebook-engineering/scaling-apache-giraph-to-a-trillion-edges/10151617006153920.

[3] A. E. Sariyuce, K. Kaya, E. Saule, and U. V. Oatalyurek, "Graph manipulations for fast centrality computation," ACM Trans. Knowl. Discov. Data, vol. 11, no. 3, pp. 26:1-26:25, Mar. 2017.

[4] A. Farooq, G. J. Joyia, M. Uzair, and U. Akram, Detection of influential nodes using social networks analysis based on network metrics, in 2018 International Conference on Computing, Mathematics and Engineering Technologies, 2018, pp. 1-6.

[5] A. Leist, and A. Gilman, "A comparative analysis of parallel programming models for c++", In The 9th International Multi-Conference on Computing in the Global Information Technology, 2014, pp. 121-127.

[6] A. Mahmoody, C. E. Tsourakakis, and E. Upfal, "Scalable betweenness centrality maximization via sampling", In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1765-1773.

[7] A. McLaughlin, and D. A. Bader, "Accelerating gpu betweenness centrality", Communications of the ACM, 2018, Vol. 61, No. 8, pp. 85-92.

[8] G. Rossetti, L. Mill, S. Rinzivillo, A. Sirbu, D. Pedreschi, and F. Giannotti, "NDlib: a python library to model and analyze diffusion processes over complex networks", International Journal of Data Science and Analytics, 2018, 5(1), 61-79.

[9] H. Ortega-Arranz, D. R. Llanos, and A. Gonzalez-Escribano, "The Shortest-Path Problem: Analysis and Comparison of Methods", Morgan and Claypool Publishers, 2014.

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

[11] J. Wei, K. Chen, Y. Zhou, Q. Zhou, and J. He, "Benchmarking of distributed computing engines spark and graphlab for big data analytics", in 2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService), 2016, pp. 10-13.

[12] L. C. Freeman, A set of measures of centrality based on betweenness, Sociometry, 1977, Vol. 40, No. 1, pp. 35-41.

[13] L. Zhang, M. Luo, and R. J. Boncella, "Product information diffusion in a social network", Electronic Commerce Research, 2020, 20(1), 3-19.

[14] M. Adler, and M. Mitzenmacher, "Towards compressing web graphs", Proceedings in DCC 2001: Data Compression Conference, 2001, pp: 203-212.

[15] M. Bernaschi, G. Carbone, and F. Vella, "Scalable betweenness centrality on multi-gpusystems", In Proceedings of the ACM International Conference on Computing Frontiers, 2016, pp. 29-36.

[16] M. Frigo, P. Halpern, C. E. Leiserson, and S. Lewin-Berlin, "Reducers and other Cilk++ hyperobjects", In Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures, 2009, pp. 79-90.

[17] M. Riondato, and E. M. Kornaropoulos, "Fast approximation of betweenness centrality through sampling", Data Mining and Knowledge Discovery, 2016, Vol. 30, pp. 438-475.

[18] M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, "Data structures and algorithms in Java", John Wiley and Sons, 2004.

[19] O. Hinz, B. Skiera, C. Barrot, and J. U. Becker, "Seeding strategies for viral marketing: An empirical comparison", Journal of Marketing, 2011, 75(6), 55-71.

[20] P. H. Du, H. D. Pham, and N. H. Nguyen, "An Efficient Parallel Method for Optimizing Concurrent Operations on Social Networks", Transactions on Computational Collective Intelligence XXIX. Lecture Notes in Computer Science, vol 10840. Springer, Cham 978-3-319-90286-9, 2018.

[21] P. H. Du, N. S. Duong, N. C. Nguyen, and N. H. Nguyen, "A Fast Computation of Betweenness Centrality in Large-Scale Unweighted Graphs", International Journal on Emerging Technologies, 2020, Vol. 11, No. 2, pp. 370-377.

[22] P. Kumar, and A. Sinha, "Information diffusion modeling and analysis for socially interacting networks", Social Network Analysis and Mining, 2021, 11(1), 1-18.

[23] P. N. Howard, A. Duffy, D. Freelon, M. M. Hussain, W. Mari, and M. Maziad, "Opening closed regimes: what was the role of social media during the Arab Spring?", Information Technology and Political Islam, 2011, pp. 1-30.

[24] R. Alhajj, and J. Rokne, Encyclopedia of Social Network Analysis and Mining, Springer, 2014, 1st edition.

[25] R. Fan, K. Xu, and J. Zhao, "A gpu-based solution for fast calculation of the betweenness centrality in large weighted networks", PeerJ Computer Science, 2017, 3.e140.

[26] R. J. Trudeau, "Introduction to Graph Theory", Dover Publications, 1994, 2 edition.

[27] R. J. Wilson, "Introduction to Graph Theory", Pearson Publisher, 2010, 5 edition.

[28] R. Puzis, Y. Elovici, P. Zilberman, S. Dolev, and U. Brandes, "Topology manipulations for speeding betweenness centrality computation," Journal of Complex Networks, 2015, vol. 3, no. 1, pp. 84-112.

[29] S. Das, "Efficient algorithms for analyzing large scale network dynamics: Centrality, community and predictability", PhD thesis, Missouri University of Science and Technology, 2018.

[30] S. Even, "Graph Algorithms", Cambridge University Press, 2011, 2nd edition.

[31] S. Priyanta, I. N. P. Trisna, and N. Prayana, "Social Network Analysis of Twitter to Identify Issuer of Topic using PageRank" International Journal of Advanced Computer Science and Applications(IJACSA), 10(1), 2019.

[32] S. Saraswathi, A. Mukhopadhyay, H. Shah, and T. Ranganath, "Social network analysis of COVID-19 transmission in Karnataka, India", Epidemiology and Infection, 2020, 148, E230. doi:10.1017/S095026882000223X

[33] T. Feder, and R. Motwani, Clique Partitions, "Graph Compression and Speeding-Up Algorithms", Journal of Computer and System Sciences, 1995, Vol. 51, Issue 2, pp. 261-272.

[34] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C.Stein, Introduction to Algorithms, 3rd-edition. MIT Press and McGraw-Hill, 2009.

[35] U. Brandes, A faster algorithm for betweenness centrality, The Journal of Mathematical Sociology, 2001, Vol. 25, No. 2, pp. 163-177.

[36] W. M. AlShammari, and M. J. Alenazi, "Performance Analysis of a Graph-Theoretic Load Balancing Method for Data Centers" International Journal of Advanced Computer Science and Applications(IJACSA), 11(8), 2020.

[37] Y. Zhang, J. Tang, Z. Yang, J. Pei, and P. S. Yu, "Cosnet: Connecting heterogeneous social networks with local and global consistency", In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 1485-1494.