A Parallel Community Detection Algorithm for Big Social Networks

Yathrib AlQahtani
College of Computer and Information Sciences
King Saud University
Collage of Computing and Informatics
Saudi Electronic University, Riyadh, Saudi Arabia

Mourad Ykhlef
College of Computer and Information Sciences
King Saud University
Riyadh,
Saudi Arabia

***Abstract*—Mining social networks has become an important task in data mining field, which describes users and their roles and relationships in social networks. Processing social networks with graph algorithms is the source for discovering many features. The most important algorithms applied to social networks are community detection algorithms. Communities of social networks are groups of people sharing common interests or activities. DenGraph is one of the density-based algorithms that used to find clusters of arbitrary shapes based on users' interactions in social networks. However, because of the rapidly growing size of social networks, it is impossible to process a huge graph on a single machine in an acceptable level of execution. In this article, DenGraph algorithm has been redesigned to work in distributed computing environment. We proposed ParaDengraph Algorithm based on Pregel parallel model for large graph processing.**

***Keywords*—Data mining; social networks; community detection; distributed computing; Pregel**

I. Introduction

Social networks have become extremely popular in the last years, and they have important roles in the dissemination of information and innovation. The analysis of such networks attracted more attention in the research area. Social networks are modeled as graphs, also called social graphs. An important property of social networks is that they have communities of entities with strong connections. Communities of social networks are groups of people sharing common interests or activities [1]. The typical way to identify communities is graph clustering.

The main techniques of graph clustering are hierarchical clustering, partitioning and density-based clustering. Hierarchical clustering algorithms, such as Newman-Girvan algorithm [2], detect several levels of clusters, where small clusters are included within the large ones. Partitioning algorithms, such as K-mean [3], divide the graph into k clusters, where k is predefined to the algorithm. Density-based algorithms consider a graph as areas of high density (clusters), surrounded by some areas of low density (noise). Not only clusters of arbitrary shape can be discovered, but also outliers and noise [4]. This capability makes density-based clustering more appropriate for social networks analysis, since usually there are a very high number of active users, but there is also a high number of users that do not contribute and can result in noise.

DenGraph [5] is a density-based clustering algorithm for community detection in social networks, inspired by the well-known clustering algorithm for spatial data, DBSCAN [6]. The main idea of DenGraph is to find clusters and outliers of weighted social networks, based on the interaction. It requires two parameters: epsilon ε, which is the maximum distance threshold; and η, the minimum number of nodes in the ε-neighborhood.

However, processing big data such as social networks with millions of vertices and edges by using conventional computation is infeasible, since it is impossible to process a huge graph on a single machine in an acceptable time. Therefore, adapting parallel computing for mining social networks has become an urgent need to address processing massive data.

In this research article, we perform DenGraph algorithm for mining implicit social graphs in distributed environment. Therefore, we propose a parallel density based clustering algorithm for social networks (ParaDengraph) in Pregel [7] model as follows. First, compute the ε-neighborhood to determine core and non-core nodes. Then, generate a new graph of the core nodes. Then, clusters are identified by finding connected components in the core graph. Finally, expand clusters with the non-core nodes.

The article is organized as follows. Section II reviews the related work. ParaDengraph algorithm is then presented in Section III. Finally, ParaDengraph was tested using real social networks, where experiments and evaluation are presented in Sections IV and V.

II. Backgraound and Related Work

Density-based algorithms consider a graph as areas of high density (clusters), surrounded by some areas of low density (noise). According to the graph, the algorithm reveals the number of clusters. Not only clusters of arbitrary shape can be discovered, but also outliers and noise. Density-based clustering requires some parameters, and generates clusters such that each cluster is a maximal set of density-connected points. Points that are not contained in any cluster are considered as noise.

A. DenGraph

Given a graph G = (V,E) consisting of a set of nodes V and a set of weighted undirected edges E, DenGraph [5] algorithm produces clusters $\{C_1,..,C_k\}$ and noise nodes, that are not part of any cluster. Other non-noise nodes are either core nodes or border nodes. A node $u \in V$ is considered as core node if it has an ε-neighborhood $N_\varepsilon(u)=\{v \in V \mid \exists(u, v) \in E \wedge dist(u, v) \leq \varepsilon\}$ of at least η neighbors ($|N_\varepsilon(u)| \geq \eta$). A Node that is non-core and connected to at least one core node is considered as border node. A core node along with its border nodes begin a cluster that can be expanded later.

For undirected and weighted graph G = (V, E), the number of interactions between two actors reflects the closeness of them, so the distance between two actors, p and q, is defined as:

$$dist(p,q) = \begin{cases} 0, & p = q \\ \min(I_{pq}, I_{qp})^{-1}, & (I_{pq} > 1) \wedge (I_{qp} > 1) \\ 1, & otherwise \end{cases} \quad (1)$$

Where I_{pq}, I_{qp} are the numbers of interactions between actors p and q initiated by p and q, respectively. The actual cluster criterion is based on the concepts: directly density-reachable, density-reachable and density-connected, which are shown in Fig. 1. These three concepts are defined as follows:

- **Definition 1**. Let u, v ∈ V be two nodes. u is directly density-reachable from v within V with respect to ε and η if and only if v is a core node and u is in its ε-neighborhood, i.e. u ∈ Nε (v).
- **Definition 2.** Let u, v ∈V be two nodes. u is density-reachable from v within V with respect to ε and η if there is a chain of nodes $p_1, \ldots, p_n$ such that p_1 = v, p_n = u and for each i = 2, . . . ,n it holds that p_i is directly density-reachable from p_{i-1} within V with respect to ε and η.
- **Definition 3.** Let u, v ∈ V be two nodes. u is density-connected to v within V with respect to ε and η if and only if there is a node m ∈ V such that u is density-reachable from m and v is density-reachable from m.

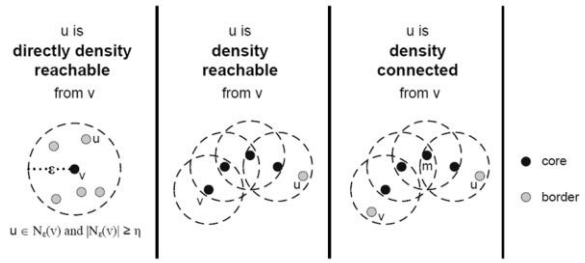

Fig. 1. Density reachability concepts.

In general, a set of core and border nodes V_C forms a cluster C if each node $u \in V_C$ is density-connected to each node $v \in V_C$. It is usually that an actor in a social network might be part of more than one community. This overlapping was not allowed in the first version of DenGraph algorithm. An extended version, called DenGraph-O, has addressed this drawback by allowing border nodes to belong to more than one cluster and it is described in Table I.

B. Graph Parallel Module: Pregel

Graphs are completely data-driven computations, dictated by vertices and edges structure rather than directly expressed in code. In addition, because of the irregular structure of graph data, scalability can be quite limited by unbalanced computational loads [8].

The Bulk Synchronous Parallel (BSP) model [9] provides the means to design parallel processing algorithms. It addresses the problem of parallelizing tasks across multiple workers by using a message-passing interface (MPI) instead of a shared memory.

Pregel introduced originally by Google [7], addresses distributed processing of large-scale graphs. It is a vertex-centric approach, where user focuses on the local action, processes each item independently, and then the system processes all actions on the large dataset.

TABLE I. DENGRAPH ALGORITHM

DenGraph

input : Graph, ε, μ
output: Overlapped clusters, noise.

```
1.  Begin
2.    Repeat
3.      Select a u ∈ V that is not yet labeled
4.      Compute ε-neighborhood(u)
5.      If u is core vertex then
6.        A new cluster id is generated
7.        u is assigned to the cluster and labeled as "core
          vertex"
8.        All v ∈ ε-neighborhood(u) are labeled as "border
          vertex"
9.        The new id is added to list of cluster-ids for all v
10.       All v are pushed on a stack
11.       Repeat
12.         Pop the top vertex v of the stack
13.         Compute the ε-neighborhood of v
14.         If v is core vertex then
15.           Label v as "core vertex"
16.           For each n ∈ ε-neighborhood(v) do
17.             Add new cluster-id to list of n
18.             Label n as "border vertex"
19.             Push n on the stack
20.           End
21.         End
22.       Until (the stack is empty)
23.     End
24.     If u is not labeled then
25.       Label u as "noise vertex"
26.     End
27.   Until (all vertices in V are labeled)
28. End
```

The basic idea is that each node of the graph corresponds to a task. The node generates output messages that are destined for other nodes, and then each node processes the inputs it receives. This computation consists of a sequence of iterations,

called super-steps. During each super-step, the framework in parallel invokes a user-defined function for each vertex.

III. PARADENGRAPH

In this section, we present a parallel density based community detection algorithm in social networks based on Pregel parallel model (ParaDengraph). Given two real parameter ε and η, and undirected weighted graph represented in adjacency lists, ParaDengraph finds clusters and any possible overlapping or noise.

ParaDengraph works as follows: Firstly, cut off the edges with distance more than ε to compute the ε-neighborhood Nε(u) for each node u, and classify it as core or non-core node. Then, generate a new graph of the core nodes only, and use this graph in Pregel model to find all the connected components in the core graph, each of which is a cluster. Finally, process non-core nodes by assigning each non-core node to the cluster(s) of its adjacent core nodes if exist, or mark it as a noise if no core node is adjacent to it. Notice that all steps are executed in parallel. ParaDengraph is divided into the following three stages:

A. Computing ε-neighborhood

ParaDengraph uses a graph parallel model and generate a graph structure from the input adjacency lists. Therefore, many graph functions can simplify the process, such as filtering feature. This step is accomplished by filtering the graph from all edges with distance > ε. The number of the remaining edges that are adjacent to each node is used to classify it.

B. Core Graph Connected Components

Core graph is generated by filtering the graph of the previous step from all nodes with adjacent edges < η. Then, the core graph is processed in Pregel model to find connected components. The connected core nodes form a cluster. Solving such problem requires unknown number of iterations (super-steps) to find all the connected core nodes by passing messages.

C. Expanding Clusters

After discovering all connected core nodes and generating all initial clusters, the next step is to process non-core nodes, assigning each to the same cluster of its neighbor core nodes, and mark it as border. Overlapping must be considered when the border node connects to more than one cluster. Note that when a node does not adjacent to any core node (cluster), it must be marked as noise. The proposed algorithm of ParaDengraph is shown in Table II.

IV. EXPERIMENTS

ParaDengraph was tested with two real networks. The whole workflow is shown in Fig. 2. ParaDengraph was implemented with Apache Spark GraphX engine for large-scale graph processing and Hadoop distributed file system (HDFS), and was run on a 64-bit PC with Intel® Core™ i5-3337U CPU @ 1.80GHz × 4, 5.7 GB RAM and 732.0 GB HD.

A. Enron Emails Network

The Enron email network [10] consists of 1,148,072 emails sent between 87,273 employees of Enron. Nodes in the network are individual employees, and edges are individual emails. We run ParaDengraph for many times, changing both parameters: epsilon ε and minimum point η, as shown in Table III. Fig. 3 shows the relation between the values of ParaDengraph parameters and the number of generated clusters of Enron network.

TABLE II. PARADENGRAPH ALGORITHM

ParaDengraph

input : Graph adjacency lists, ε, μ
output : Overlapped clusters, noise.

```
1. Begin
2.   Generate the graph (G)
3.   Filter G from any edge with distance > ε.
4.   CoreGraph=Filter G from nodes with neighbors < η.
5.   Mark each u in CoreGraph as "CORE"
6.   NoneCore_nodes=Filter G from nodes with neighbors>η.
7.   Start Pregel(CoreGraph): vertex-program
8.     Receive messages from the connected vertices.
9.     Compute label = min(u ID, u label, IDs of all source
       vertices of the received messages).
10.    If label is changed
11.    |send messages to connected vertices with the new one
12.    End
13.  Until (no new messages).
14.  Join CoreGraph and NoneCore_nodes
15.  Set for each non-core u label = list of labels of all
     directly reachable core.
16.  If u label is empty
17.  |  Mark u as "NOISE"
     Else
     |  Mark u as "BORDER"
18.  End
19. End
```

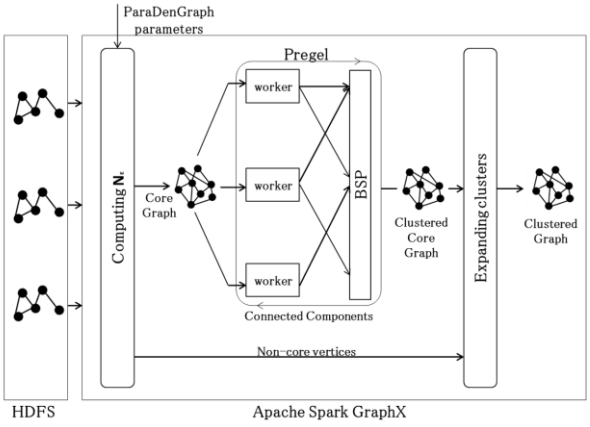

Fig. 2. ParaDengraph workflow.

TABLE III. ENRON NETWORK RUN STATISTICS

Epsilon	Min points	Clusters	Noise	Cores	Borders
0.1	1	31	86346	927	0
	2	17	86374	387	512
	3	10	86408	237	628
	4	10	86445	168	660
	5	9	86466	131	676
	6	9	86494	102	677
	7	7	86515	87	671
	8	9	86521	81	671
	9	8	86544	69	660
	10	8	86559	60	654
0.2	1	31	85494	1779	0
	2	15	85526	830	917
	3	10	85568	557	1148
	4	7	85596	409	1268
	5	5	85618	320	1335
	6	5	85647	254	1372
	7	6	85663	215	1395
	8	7	85679	191	1403
	9	6	85715	166	1392
	10	5	85750	144	1379
0.3	1	25	85068	2205	0
	2	15	85088	1028	1157
	3	11	85124	707	1442
	4	12	85149	534	1590
	5	10	85188	426	1659
	6	7	85226	342	1705
	7	5	85250	287	1736
	8	4	85279	245	1749
	9	6	85309	216	1748
	10	6	85318	193	1762
0.4	1	26	84476	2797	0
	2	13	84502	1336	1435
	3	11	84533	963	1777
	4	8	84573	740	1960
	5	7	84607	598	2068
	6	8	84630	487	2156
	7	7	84656	417	2200
	8	5	84691	358	2224
	9	4	84714	310	2249
	10	4	84725	272	2276
0.5	1	40	83327	3946	0
	2	12	83383	1902	1988
	3	11	83426	1406	2441
	4	8	83471	1103	2699
	5	6	83509	908	2856
	6	6	83538	748	2987
	7	6	83568	648	3057
	8	5	83599	567	3107
	9	2	83647	491	3135
	10	2	83663	441	3169

B. Twitter Interactions Network

The second data set was Twitter interaction graph, where nodes represent users and edges represent interactions, such as retweets or replays. ParaDengraph was applied many times to the graph of 9187 nodes as shown in Table IV. The relation between the generated clusters of Twitter graph and the parameters values of ParaDengraph is shown in Fig. 4.

From both Fig. 3 and 4, it can be seen that the resulted clusters were not related directly to the parameters values, since the number of clusters decreased in Enron network with increasing the values, while it increased in the Twitter network. Therefore, the result depends mainly on the graph nature and the core nodes locations to each other's.

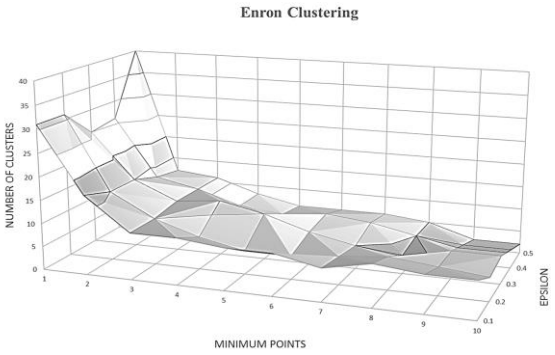

Fig. 3. Enron clusters and parameters values.

TABLE IV. TWITTER NETWORK RUN STATISTICS

Epsilon	Min points	Clusters	Noise	Cores	Borders
0.1	1	9	8792	395	0
	2	7	8796	36	355
	3	9	8796	16	375
	4	9	8796	15	376
	5	9	8801	13	373
	6	8	8807	12	368
	7	8	8807	12	368
	8	8	8807	12	368
	9	8	8807	12	368
	10	8	8807	12	368
0.2	1	5	8377	810	0
	2	4	8379	70	738
	3	8	8379	29	779
	4	7	8385	19	783
	5	7	8385	18	784
	6	7	8385	17	785
	7	7	8385	16	786
	8	7	8385	16	786
	9	7	8385	16	786
	10	7	8385	16	786
0.3	1	6	8176	1011	0
	2	3	8182	85	920
	3	8	8182	31	974
	4	8	8182	21	984
	5	7	8187	20	980
	6	6	8191	18	978
	7	7	8191	16	980
	8	7	8191	16	980
	9	7	8191	16	980
	10	7	8191	16	980
0.4	1	4	7756	1431	0
	2	3	7758	111	1318
	3	6	7759	44	1384
	4	8	7763	26	1398
	5	8	7763	22	1402
	6	6	7772	19	1396
	7	6	7772	18	1397
	8	5	7780	17	1390
	9	6	7780	16	1391
	10	6	7780	16	1391
0.5	1	3	6787	2400	0
	2	3	6787	182	2218
	3	5	6787	61	2339
	4	5	6787	36	2364
	5	9	6792	25	2370
	6	9	6792	23	2372
	7	9	6792	22	2373
	8	8	6799	20	2368
	9	8	6799	20	2368
	10	8	6799	20	2368

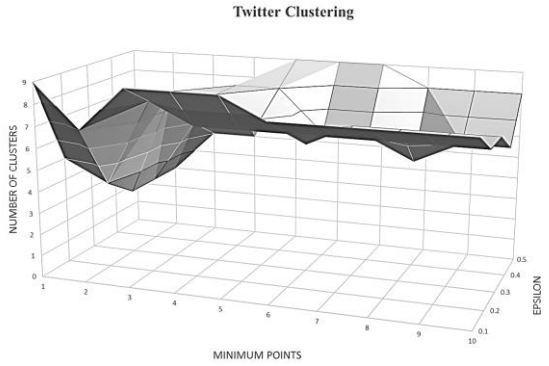

Fig. 4. Twitter clusters and parameters values.

However, the clear result was the relation between the number of core nodes and the parameters values. The number of core nodes increased when the epsilon value increased, but decreased with increasing the value of minimum points.

V. Evaluation

For evaluation purpose, both ParaDengraph and Newman Edge Betweenness Clustering (EBC) [2] algorithms were applied to the same social network graph to compare both results and obtain the characteristics of both clustering methods. The previous Enron and Twitter graphs, with 87,273 and 9,187 nodes respectively, are too large to fit in the low capacity of EBC. So, for evaluation, a smaller Twitter data set (539 nodes) was used to run both algorithms. ParaDengraph was applied on the graph with different collections of parameters values, epsilon (ε) and minimum points (η). Since the average noise percentage for all runs (140 runs) was 54.24 %, we chose the closer epsilon value (ε=0.09). The result with ε=0.09 and η=5 was 8 clusters, and 294 nodes as noise.

Then, Newman EBC was applied to the same graph of 539 nodes. Notice that EBC is a hierarchical clustering algorithm. EBC gave the best modularity at level 9 with 17 clusters. While EBC generated 17 clusters, ParaDengraph with the ability of density-based clustering to discover noise generated only 8 clusters. In order to test the effect of noise in the result, a total of 294 noise nodes that discovered by ParaDengraph (ε= 0.09 and η=5) were removed from the original graph, and EBC was applied again. On the graph of 245 nodes (after removing noise), the best modularity was found at level 6 with 10 clusters. The number of clusters decreased from 17 to 10, which was closer to ParaDengraph result (8 clusters).

As a result, we found that ParaDengraph outperformed EBC mainly in two parts. First, EBC failed to deal with large social networks due to its high time complexity. However, ParaDengraph was capable to run with better performance, even in the sequential version. This result proves that density-based clustering algorithms are more appropriate for large social networks than hieratical clustering algorithms. Second, notice that EBC at the first run on the complete graph (539 nodes) gave a result of 17 clusters. However, the result of the second run on the same graph after excluding all noise (245 nodes) was 10 clusters. In the other hand, ParaDengraph gave both 8 clusters and some noise. We can notice that this result (8 clusters) was too close to the second result of EBC (10 clusters) after removing all noise.

VI. Discussion

In addition to the suitability for large-scale graph computing, which is most critical in dealing with massive networks, ParaDengraph is also able to give reasonable clusters by discovering and excluding noise from the clustering process. This feature is essential in social networks, since there is usually a large number of actors, but there is also a high number of them do not contribute (outliers). While EBC found clusters, it was unable to detect outliers, resulting in more clusters where some may contained only one or very few nodes. However, outliers did not affect the outcome of ParaDengraph.

Moving to the limitations of this proposed algorithm, the most noticeable limitation is that ParaDengraph was proposed based on the static version of DenGraph, where communities are observed as static. However, they are evolving continuously and such dynamic networks require considering the time and changes over time.

VII. Conclusion

This article proposed ParaDengraph, a graph-parallel algorithm for community detection in large social networks based on the original sequential algorithm called DenGraph. The suggested algorithm is suitable for large-scale graph computing. ParaDengraph has been applied to two real social networks: Enron emails and Twitter. For evaluation, ParaDengraph was compared with Newman Edge Betweenness Clustering (EBC) algorithm. ParaDengraph outperformed EBC in terms of performance and the ability to generate clusters that are more reasonable by excluding noise from the clustering process. For future work, ParaDengraph can be improved for studying communities on dynamic social networks.

Acknowledgments

This work was supported by the Deanship of the Scientific Research and Research Center of the Collage of Computer and Information Sciences, King Saud University.

References

[1] Zafaran, R., Abbasi, M. A., & Liu, H. (2014). “Social media mining, an introduction” Cambridge University Press.

[2] Newman, M., & Girvan, M. (2004). “Finding and evaluating community structure in networks”. Phys. Rev. E, pp. 69, 026113.

[3] MacQueen, J. (1967). “Some methods for classification and analysis of multivariate observations”. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, University of California Press.

[4] Subramani, K., Velkov, A., Ntoutsi, I., Kroger, P., & Kriegel. (2011). “Density-based community detection in social networks”. Internet Multimedia Systems Architecture and Application (IMSAA) (pp. 1-8). IEEE 5th International Conference.

[5] Falkowski, T., Barth, A., & Spiliopoulou, M. (2007). “DENGRAPH: a density-based community detection algorithm”. Web Intelligence (pp. 112-115). IEEE/WIC/ACM International Conference.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", In: Proc. Of KDD, 226-231, 1996.

[7] G. Malewicz, M. H. (2010). "Pregel: a system for large-scale graph processing". ACM SIGMOD International Conference on Management of data (pp. 135-146). New York, NY, USA: ACM.

[8] Lumsdaine, A., Gregor, D., Hendrickson, B., & Berry, J. (2007, 5 17). "Challenges in parallel graph processing". Parallel Processing Letters, pp. 5–20.

[9] Leslie G. Valiant, "A bridging model for parallel computation". Comm. ACM 33(8), 1990, 103–111.

[10] W. Cohen, Enron network dataset, KONECT. (2017)