An Efficient Distributed Subgraph Mining Algorithm in Extreme Large Graphs

Bin Wu and YunLong Bai

School of Computer Science Beijing University of Posts and Telecommunications
Beijing 100876, China
wubin@bupt.edu.cn, baiyunl303@163.com

Abstract. Graph mining plays an important part in the researches of data mining, and it is widely used in biology, physics, telecommunications and Internet in recently emerging network science. Subgraph mining is a main task in this area, and it has attracted much interest. However, with the growth of graph datasets, most of these former works which mainly rely on single chip computational capacity, cannot process massive graphs. In this paper, we propose a distributed method in solving subgraph mining problems with the help of MapReduce, which is an efficient method of computing. The candidate subgraphs are reduced efficiently according to the degrees of nodes in graphs. The results of our research show that the algorithm is efficient and scalable, and it is a better solution of subgraph mining in extreme large graphs.

Keywords: subgraph mining; large graph; MapReduce; distributed algorithm.

1 Introduction

Graph-structured data is widely used to represent complicated structures, and it is becoming increasingly abundant in many application domains [4]. The whole society can be represented by a topological graph, in which every person is a node and the relationship of two persons is an edge. Therefore, society can be studied through the research of graph mining. The biologists have found that researches of gene coordination are difficult in biology genetics, which is mainly caused by the changeful reaction between two different genes. A large amount of researches must be done to find out the gene structure's reaction to medicine. However, these kinds of researches are expensive and time consuming. So a befitting method to define the structures of genes is desired. Therefore, the protein structure can be described as a graph structure, in which atoms are the vertices of the figure, while the price of atoms is the corresponding edge. The intrinsic relationship and shared mode between proteins can be dug through graph mining, and these shared patterns can guide the gene experiments in turn. Because of the urgent requirements of these applications, graph mining becomes an important research field, and these researches have brought applications.

In order to discover some useful knowledge from graph-structured databases, graph mining [12], [13], especially subgraph discovery, has been studied intensively. Among most of the applications in graph mining, subgraph is a well-studied problem, since it has a wide and constantly expanding range of applications areas, which include biochemistry, web mining, program flow analysis, etc. Moreover, in the

F.L. Wang et al. (Eds.): AICI 2010, Part I, LNAI 6319, pp. 107–115, 2010.

VAST 09 Flitter Mini Challenge the user-defined criminal structure is identified within a social network based on two possible hypotheses. The hypothesis most closely matched in the network represents the criminal social structure [17]. As a consequence, several subgraph mining algorithms have been developed. Some of them rely on principles from inductive logic programming and describe the graph structure by logical expressions [14]. However, the vast majority transfers techniques that were originally developed for frequent item set mining. Examples include MolFea [15], FSG [16], MoSS/MoFa [8], FFSM [9], Gaston [10], gSpan [2] and CloseGraph [5]. Besides those locally optimized algorithms, there are also some parallel or distributed solutions, such as [6]. However, most of these works are based on theoretical distributed methods and do not provide systematic ones. "Cloud" based computing on large scale data has aroused great interests in both industry and research areas [3]. Moreover, recently Google's MapReduce [11] has largely simplified the distributed computing process, without considering any underlying complexity. Meanwhile, MapReduce computational model is also widely applied beyond the cloud as well [1] [4]. Although recent works have focused on graph mining using MapReduce, they are generally quite simple, such as PageRank, or avoiding intractable problems, such as clique enumeration.

In this paper, we provide a parallel subgraph mining algorithm in large graphs. To use the algorithm, the diameter of the motif to be matched must be calculated. Then, we can decide how to represent the large graph. The candidate subgraphs are reduced efficiently according to the degrees of vertexes in graphs. The algorithm is quite efficient and scalable comparing with prior algorithms. This paper is organized as follows: Section 2 presents a short description of preliminaries. Section 3 describes the algorithms which are implemented with MapReduce structure. In section 4, we give an evaluation of our algorithms. Finally, we conclude our work in Section 5.

2 Notation and Definition

Given a graph G, V (G) represents its vertices and E (G) represents its edges. In this paper, we assume without loss of generality that G is simple.

$$\Gamma(v) = \{u \in V(G) \mid (u, v) \in E(G)\}$$

Γ(v) represent the neighbours of v, and P(v, u) represent the shortest path length between v and vertex u.

$$\Gamma_N(v) = \{u \in V(G) \mid P(v, u) \leq N\}$$

$\Gamma_N(v)$ represents the "N-leap" neighbours of v, and $k_v = |\Gamma(v)|$ represents the degree of v.

Definition 1. [Network Diameter] In an undirected graph G, the diameter is the maximum length of all shortest paths. D (G) represents the diameter of G.

Definition 2. [Personal Centre Network] v is a vertex of graph G, the edges between v and its neighbours Γ (v) are the Personal Centre Network of node v. We use C(v) to represent it.

Definition 3. [Local Degree] The degree of a vertex v in a subgraph of G is a local degree of node v. One vertex contains different Local Degree in different subgraph of G. Local Degree is represented by LD(v).

Definition 4. [Subgraph Mining] Suppose that the input database is

$$GD=\{G_i \mid i = 0, 1, 2, \ldots. n\}.$$

If subgraph g is isomorphism with G_i, output G_i.

3 Subgraph Mining Algorithm

First and foremost, we describe the algorithm of subgraph mining when the subgraph's network diameter is 2. Before describing the algorithm, we must represent the large graph as Personal Centre Network. Then we summed up the general parallel algorithm of subgraph mining. All the algorithms are implemented with MapReduce structure.

3.1 Enumerating Personal Centre Networks

We usually represent a graph as the adjacency lists of all nodes, and each adjacency list is the information of a node and its neighbours. Here, in order to dig subgraph, we represent the graph as personal centre network. The following shows a parallel process of transforming representation. To the beginning, we should obtain the "two-leap" information of a specific vertex in the graph, namely v, Γ(v) and Γ(Γ(v)) (v's neighbours' neighbours)[1]. Taking the graph in Fig. 1 for example, to find the personal centre network of vertex 1, we can first get < 1, < 2, 3, 4 >>, then < 2, < 3, 4 >> and < 3, 4 > from the adjacency list.

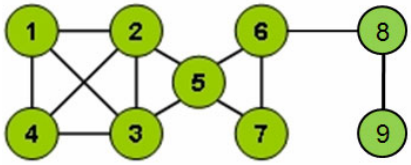

Fig. 1. A simple graph

To reserve the "two-leap" information of a vertex, we present our method, which can also be viewed as subgraph generation based on adjacency list [4]. Let < 1, < 2, 3, 4 >>be a record in the adjacency list, we transform it into the records:

< 2, < 1, <2, 3, 4 >>>
< 3, < 1, < 2, 3, 4 >>>
< 4, < 1, < 2, 3, 4 >>>

For other records in adjacency list, we perform the same operation and finally generate an intermediate result in which each record contains a vertex v, one of its neighbours v' and Γ(v'). Considering in MapReduce, we present this transformation in a Mapper as in Table 1.

Thus far, given a graph, we transform it into a formation, in which each record conveys a vertex and part of its "two-leap" information. So, we can collect all the "two-leap" information of a vertex and the personal centre network of the vertex can

Table 1. Personal Centre Network

Personal Centre Network	
Map	Input：adjacent list Key：line position Value：<k, $\Gamma(k)$>
	For each v in $\Gamma(k)$ output <v, <k, $\Gamma(k)$>> output <k, $\Gamma(k)$>
	Output：<v, <k, $\Gamma(k)$>> and <k, $\Gamma(k)$>
Reduce	Input： Key：v Value：list<v', $\Gamma(v')$>和$\Gamma(v)$
	For each vertex vv in $\Gamma(v)$ If vv∈list< v', $\Gamma(v')$ Output <vv , v'>
	Output：Graph record represented as Personal Centre Network

be represented. For example, given a vertex 4, we collect all the records of its "two-leap" records as follows:

< 4, < 1, < 2, 3, 4 >>>
< 4, < 2, < 1, 3, 4, 5 >>>
< 4, < 3, <1, 2, 4, 5 >>>

We can get the Personal Centre Network of vertex 4 is

<4, <<1, 2>, <1, 3>, <2, 3>>>

The whole algorithm is described in Table 1.

3.2 Mining Subgraphs(Whose Diameter Is 2)

Given a certain subgraph named motif as is shown in Figure 2. We can recognize that the network diameter of motif is 2.

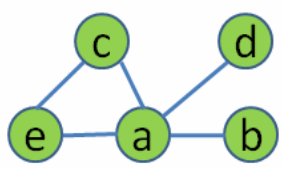

Fig. 2. Motif

At the beginning, we must calculate the subgraph's basic information content, which includes node number and sorted degree list. So, the basic information of motif is as follows: the number of vertex is 5 and the sorted degree list D(m)={4, 2, 2, 1, 1}. We can start the subgraph mining algorithm described below.

First we represent the large graph as Personal Centre Network of every vertex, which is described at section 3.1. Then make the following address of each record (represented as C(v)) of the large graph. We take figure 2 for an example, mining subgraph in figure 1.

Step 1: If the number of vertex in C(v) is less than n, discard C(v). Or, calculate the local degree of all nodes in C(v), and then sort them as a list d(C(v)). For example: the Personal Centre Network of vertex 4 in figure 1 is C(4)=<4, <<1, 2>, <1, 3>, <2, 3>>>, the sorted local degree is LD(C(4))={3, 3, 2, 2}; the Personal Centre Network of vertex 2 is C(2)=<2, <<1, 3>, <1, 4>, <3, 4>, <3, 5>>>, and its sorted local degree is LD(C(2))={4, 4, 3, 3, 2}.

Step 2: According to the sorted local degree list LD(C(v)), obtained by step 1, we select the previous n numbers to compare with D(m). Delete C(v), if LD(C(v))(v) < D(m)(i) ($0 \le i < n$) at the corresponding position. Such as LD(C(4)) and D(m). Otherwise, do the next step.

Step 3: if there has a vertex whose local degree is less than the smallest number in d(m), then delete the vertex and all edges connected to it in C(v).

Step 4: After the first three steps, we calculate all the n combinations of all remaining nodes in C(v). Notice that each of the combination must contain node v. Then, output the combinations and edges between them as candidate subgraphs. For example, the candidate subgraph of C(2) is <2, <<1, 3>,<1, 4>,<3, 4><3, 5>>>.

Step 5: Calculating the candidate subgraph and motif according to graph isomorphism algorithms. We output the candidate subgraphs if they are isomorphic with motif.

We can calculate the sorted local degree list of each candidate subgraph which is the output of Map, and represent the list as LD(c). If LD(c) equals with the motif degree list D(m), continue the next step of the algorithm; otherwise discard the candidate subgraph. Therefore, the efficiency of the algorithm can be improved.

3.3 General Algorithm of Subgraph Mining

In this section, we will give a summary of the general algorithm of subgraph mining. First of all, we should calculate the basic information content of motif, which includes the network diameter D(m), the vertex number N and the sorted degree list L(m). Secondly, transform the representation of large graph to $G_{D(m)}$. Each record $\Gamma_{D(m)}(v)$ of $G_{D(m)}$, which is the "N-leap" neighbours of a vertex v, is processed by the following steps:

Step 1: If the number of vertex in $\Gamma_{D(m)}(v)$ is less than N, discard $\Gamma_{D(m)}(v)$ .Otherwise, calculate the local degree of all nodes in $\Gamma_{D(m)}(v)$. If there has a vertex whose local degree is less than the smallest number in D(m), then delete the vertex and all edges connected with it in $\Gamma_{D(m)}(v)$. The remaining subgraph is represented as $\Gamma'_{D(m)}(v)$.

Table 2. Distributed Subgraph Mining Algorithm

Subgraph Mining	
Map	Input1 : Graph file (graph is represented as "D(m)-leap" of all vertexes) Key : line position Value : $\Gamma_{D(m)}(v)$ =<vertex1, <vertex2, vertex3>…….> Input2 : Basic information of motif. The number of nodes N and the sorted degree list D(m).

	For each vertex V_1 in $\Gamma_{D(m)}(v)$
	If(LD(V_1)<D(m)(N))
	Delete V_1
	For (i=0;i<N;i++)
	If (LD($\Gamma_{D(m)}(v)$)(i) < D(m)(i))
	Delete $\Gamma_{D(m)}(v)$
	If the remaining vertex number of $\Gamma_{D(m)}(v)$ is smaller than N, delete $\Gamma_{D(m)}(v)$. Sorting the local degree of remaining vertexes in $\Gamma_{D(m)}(v)$, representing as LD($\Gamma_{D(m)}(v)$).

	Output: Candidate subgraphs(vertex v, its n-1 neighbors and all edges between them.
Reduce	Input1 : Candidate subgraphs Input2 : motif
	Calculating the candidate subgraph and motif according to graph isomorphism algorithms. output the candidate subgraphs if they are isomorphic with motif
	Subgraphs which are isomorphic with motif

Step 2: Keep on doing step 1 until there is no vertex to be deleted. Then sort the vertexes' local degrees of remaining subgraph as a list LD ($\Gamma'_{D(m)}(v)$).

Step 3: According to the sorted local degree list LD ($\Gamma'_{D(m)}(v)$), obtained by step 2, we select the previous n numbers to compare with D (m). Delete $\Gamma'_{D(m)}(v)$, if

$$\text{LD}\ (\Gamma'_{D(m)}(v))\ (i) < \text{D}\ (\text{m})\ (\text{i})\ (0 \le \text{i} < \text{n})$$

at the corresponding position. If the number of remaining vertex is bigger than N, delete $\Gamma'_{D(m)}(v)$. Otherwise, the remaining vertexes and edges between them are represented as $\Gamma^{R}_{D(m)}(v)$, do step 4.

Step 4: After the first three steps, we calculate all the N combinations of the nodes in $\Gamma^{R}_{D(m)}(v)$. Notice that each of the combination must contain node v. Then output the combinations and edges between them as candidate subgraphs.

Step 5: Calculating the candidate subgraph and motif according to graph isomorphism algorithms. We output the candidate subgraphs if they are isomorphic with motif. A complete demonstration of this procedure is described in Table 2.

4 Experiments

All the algorithms are implemented in Java language. The result is obtained on a cluster environment, composed of one master node and 8 slave nodes (Intel(R) Xeon(R) CPU 2.00 GHz, Linux RH4 OS) with 500G*4*8 total storage capacity, and the cluster is deployed with Hadoop-0.20.2 platform. First we test our algorithm by datasets of social network structure in the VAST 09 Flitter Mini Challenge, and the results are the same as what we get in a stand-alone algorithm, though the parallel algorithm is inefficient.

The second experimental dataset is described as follows: it is a service meter datasets of mobile phone comes from one day's communication records in a city in China, and it includes 16,123,172 vertex and 73,636,994 edges. The diameter of the subgraph to be matched is 2 and motif contains 8 vertexes. We use the default Mapper number which is S/64M, S represents input dataset size and 64M is the default store size of each block of the dataset.

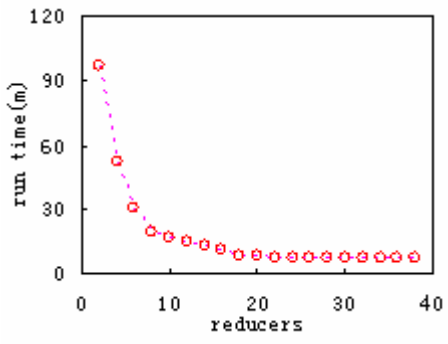

Fig. 3. Runtime with a varying number of reducers

Figure 3 shows the runtime of our algorithms on the dataset (R stands for reducer number). From Figure 3, we observe that the reducing of runtime is not constant as the same times as the increasing of reducer number. Along with increasing reducers' number from 2 to 8, the runtime approximately reduces to 1/2, 1/3 and 1/4. However from the last several number of reducer, we find that along we increase reducers' number from 18 to 64, the runtime does not decrease by times accordingly.

Theoretically the number of reducers can directly indicate the degree of concurrence. Thus the theoretical aggregated throughout versus n reducers is calculated as n×T1, where T1 is the throughout time of the single version [1]. In Figure 4, we plot experimental values in red marks and theoretical values in blue marks. We find the values present a "grow-smooth-decline" procedure. Initially the speedup of our algorithm almost coincides with that in theory. We also find that the optimum speedup is reached at from 8 to 16 reducers, which is just the number of computing nodes (8)

and CPU cores (8 × 2) of our cluster. The speedup deviates from theoretical values after the optimum value significantly. This also shows that after the optimum point, increasing of reducer number cannot get the corresponding efficiencies.

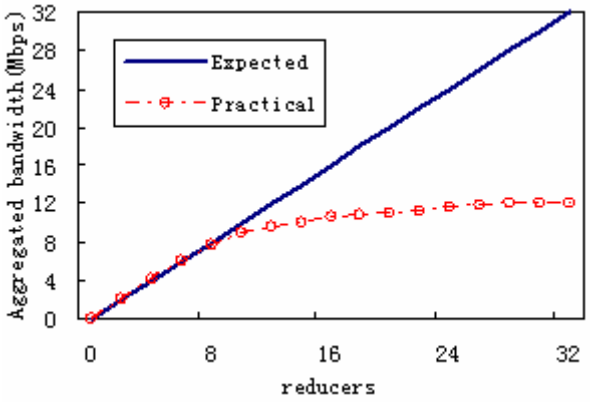

Fig. 4. Performance comparison between expectation and practice

5 Conclusions

In this paper, with the help of cloud-based MapReduce structure, we have developed a parallel algorithm that can perform subgraph mining with a better performance in both theoretical analysis and practical results. In fact, there are a lot of subgraph mining algorithms, but most of these algorithms are inefficient when dealing with extreme large graphs. The algorithm proposed in the paper has the better performance over all other algorithms in large graph mining so far.

The main contribution of the paper is summarized as follows:

1. We presented a massively parallel subgraph mining algorithm, in which the large graph is represented as "n-leap".
2. Reduce the number of candidate subgraphs by local degrees of the vertexes in each graph record.

Experiments on a massive dataset demonstrate outstanding scalability of the algorithm. Having solved the subgraph mining issue in MapReduce framework, our future work will continue to research on all kinds of graphs in real life, such as directed graph, undirected graph and weighted graph. Besides theoretical methods, we also concern the cloud-based MapReduce applications and make more practical stabs in such powerful tool.

Acknowledgement. This work is supported by the National Science Foundation of China (No.90924029, 6107412860905025), National High Technology Research and Development Program of China (No.2009AA04Z136), National Key Technology R&D Program of China 2006BAH03B05-3, 2006BAJ16B04-02) and the Fundamental Research Funds for the Central Universities.

References

[1] Yang, S., Wang, B., Zhao, H., Wu, B.: Efficient Dense Structure Mining Using MapReduce icdmw. In: IEEE International Conference on Data Mining Workshops, pp. 332–337 (2009)

[2] Yan, X., Han, J.: gSpan: Graph-Based Substructure Pattern Mining. In: Proc. 2nd IEEE Int. Conf. on Data Mining (ICDM 2003), Maebashi, Japan, pp. 721–724. IEEE Press, Piscataway (2002)

[3] Chu, C.T., Kim, S.K., Lin, Y.A., Yu, Y., Bradski, G.R., Ng, A.Y., Olukotun, K.: Map-reduce for machine learning on multicore, pp. 281–288. MIT Press, Cambridge (2006)

[4] Cohen, J.: Graph twiddling in a mapreduce world. Computing in Science and Engineering 11(4), 29–41 (2009)

[5] Yan, X., Han, J.: Closegraph: Mining Closed Frequent Graph Patterns. In: Proc. 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, KDD 2003 (2003)

[6] Du, N., Wu, B., Xu, L., Wang, B., Pei, X.: A parallel algorithm for enumerating all maximal cliques in complex network. In: ICDMW 2006, pp. 320–324 (2006)

[7] Kuramochi, M., Karypis, G.: Finding Frequent Patterns in a Large Sparse Graph. In: Proc. 4th SIAM Int. Conf. on Data Mining (SDM 2004), Lake Buena Vista, FL. Society for Industrial and Applied Mathematics, Philadelphia (2004)

[8] Borgelt, C., Berthold, M.R.: Mining Molecular Fragments: Finding Relevant Substructures of Molecules. In: Proc. IEEE Int. Conf. on Data Mining (ICDM 2002), Maebashi, Japan, pp. 51–58. IEEE Press, Piscataway (2002)

[9] Huan, J., Wang, W., Prins, J.: Efficient Mining of Frequent Subgraphs in the Presence of Isomorphism. In: Proc. 3rd IEEE Int. Conf. on Data Mining (ICDM 2003), Melbourne, FL, pp. 549–552. IEEE Press, Piscataway (2003)

[10] Nijssen, S., Kok, J.N.: A Quickstart in Frequent Structure Mining Can Make a Difference. In: Proc. 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2004), Seattle, WA, pp. 647–652. ACM Press, New York (2004)

[11] Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: OSDI 2004, pp. 137–150 (2004)

[12] Cook, D.J., Holder, L.B. (eds.): Mining Graph Data. Wiley-Interscience, Hoboken (2005)

[13] Washio, T., Motoda, H.: State of the art of graph-based data mining. SIGKDD Explorations 5(1), 59–68 (2003)

[14] Finn, P.W., Muggleton, S., Page, D., Srinivasan, A.: Pharmacore Discovery Using the Inductive Logic Programming System PROGOL. Machine Learning 30(2-3), 241–270 (1998)

[15] Kramer, S., de Raedt, L., Helma, C.: Molecular Feature Mining in HIV Data. In: Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD 2001), San Francisco, CA, pp. 136–143. ACM Press, New York (2001)

[16] Kuramochi, M., Karypis, G.: Frequent Subgraph Discovery. In: Proc. 1st IEEE Int. Conf. on Data Mining (ICDM 2001), San Jose, CA, pp. 313–320. IEEE Press, Piscataway (2001)

[17] Zhou, H., Shaverdian, A.A., Jagadish, H.V., Michailidis, G.: Multiple step social structure analysis with Cytoscape. In: IEEE Symposium on Visual Analytics Science and Technology, VAST 2009, October 12-13, pp. 263–264 (2009)