International Journal of Computer Applications (0975 – 8887)
Volume 93 – No 9, May 2014

Performance Improvement through Parallelization of Graph Clustering algorithm

Yogendra Kumar Dehariya　　Ravi Shankar Singh

Indian Institute of Technology (BHU), Varanasi 221005 India

ABSTRACT

Clustering is the task of Grouping of elements or nodes (in the case of graph) in to clusters or subgroup based on some similarity metrics. In general Clustering is unsupervised learning task requires very little or prior knowledge except the data set. However Clustering Task are computationally expensive as most of the algorithms require recursion or iterations and most of the time we have to deal with real life data set which are generally very huge in size. This paper deals with a well-known clustering algorithm MCL (Markov Clustering Algorithm) and proposes a parallel version of it using OPENMP.

Keywords

MCL (Markov Clustering Algorithm),OPEN-MP, Parallelization.
.

1. INTRODUCTION

Clustering is defined as dividing elements in to groups called Clusters[1][5]. Clustering task have been used in many fields including Image/Video Processing, Machine Learning, Data Mining, biochemistry, bioinformatics etc. Different Types of Clustering Algorithms have been developed like partitional, hierarchical, graph-based clustering etc. according to the properties of the data to be clustered. Most of the Clustering algorithm involves iterative or recursive function calls to find locally and globally optimal solution from a high dimensional Data Set [6]. Also the data sets are real life data sets which involve high computation for interpretation of clusters. Hence they are computationally expensive and saving computational complexity is a significant issue for the clustering algorithm. Therefore parallelization of clustering algorithm is very practical approach. This paper proposes a parallel implementation of MCL with OPEN-MP. The Improvement in its run Time is analyzed with real Bloggers Data and also with random Graph generated by using four different types of stochastic graph generation algorithm. MCL is based on the random walk it works by iteratively strengthening the strong currents and weakening the weak currents until convergence using expansion and inflation defined later in this paper.

2. ALGORITHM DESCRIPTION

For the analysis of MCL (Markov Clustering Algorithm)[2][3][4] graph was generated by using bloggers data and two stochastic graph generator i.e. Erdos Renyi and Barabasi-Albert algorithm. The next subsections describes MCL and theses algorithm in details.

2.1 MCL (Markov Clustering Algorithm)

The key idea behind Markov Clustering algorithm is that if a random walker start random walk from a dense cluster than he/she would likely to remain in that cluster until most of its nodes have been visited. MCL simulates this by iteratively modifying the transition probability matrix M by following to operations [3]:

- **Expansion**: it is defined as follows

M = M*M

If the expansion parameter is e than M is raised to M to the power e. which corresponds to taking e steps in random walk.

- **Inflation**: it is defined with respect to inflation parameter r in which each entry of M is raised to power r and then it is normalized.

These steps are repeated until convergence.

2.2 Algorithm used for Graph Generation

2.2.1Using Bloggers Data (Author Proposed algorithm)

As our input data, a real time blog data has been adopted from [7]. The blog data contains different tags for each blog. The graph constructed by considering each blog as a node and edge strength between them has been calculated on the basis of similarity between their tags. Let maximum number of tags can be given for a blog is ‘n’ then edge strength between two nodes can be given as ‘s/n' where s is the number of similar tags.

Pseudo code for generating the graph is given below:

```
Let n = number of maximum tags that a
author can have.
nst(B[1],B[2]) = function returning the
number of similar tags between blog B1
and blog B2
for I = 1 to n
for j = 1 to n
{
  s = nst(B[i],B[j]);
  M[i][j] = s/n;
 }
```

The above algorithm defines how the graph is generated from the blog data. It can further be optimized. The final graph generated is in the form of adjacency matrix M.

2.2.2 Using a Stochastic Graph generator

The following Two Different stochastic Graph Generator is used for performance analysis of MCL :

Barabási–Albert algorithm: It begins with a initially connected nodes of m_0 nodes. New nodes are added one at a time and is connected to m existing nodes ($m < m_0$) with a probability that is proportional to the number of links that the existing nodes already have. Formally it can be written as :-

$P_i = (K_i / \sum_j K_j)$

Where P_i = the probability that the new node is connected to node i

K_i = Degree of the node i

Erdős–Rényi Algorithm:

In this is a graph is constructed on the basis of connecting nodes randomly. An edge is included with probability p which is 0.10 in our implementation.

3. PROPOSED PARALLEL VERSION OF MCL ALGORITHM

As the two main steps involved in MCL i.e. Expansion and Inflation involves iteration which can be computed independently from each hence they can be parallelized .The parallelization is done at both the expansion and inflation steps as follows:-

Expansion:

Since expansion steps involves multiplication of transition matrix and hence the multiplication can be performed parallel as shown:-

```
For all i = 0 to n

  For all j =0 to n
    For k = 0 to n
      M*[i][j] +=  M[i][k]*M[k][j];
 M = M*;
```

Inflation:

Inflation step is nothing but each element raised to inflation parameter hence each element can be computed in parallel as follows:-

```
For all i=0 to n
    M[i][j] = M[i][J]^n;
```

Where M = transition matrix
N = Number of nodes

4. EXPERIMENTAL SETUP

The MCL algorithm was implemented in c++ using OPEN-MP as given in the above pseudo code and the improvement in its run time is observed using the graph generated from Bloggers Data and random Graph Generated using Stochastic Graph Generator.

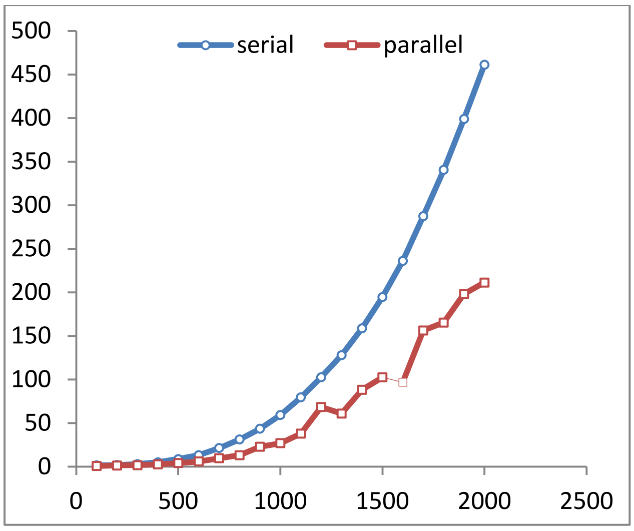
5. EXPERIMENTAL RESULT

5.1 Using Graph Generated From Bloggers Data

The observation for bloggers data graph is summarized in Table 1 and graph for the same is shown in graph 1. It can be easily evident from the Table 1 that a speed up of approximately 0.5 can be easily obtained through parallelization of MCL.

Table 1 : Observations for Bloggers Data

Serial no.	Number of nodes	Number of Clusters	Run Time (Sequential)	Run Time (Parallel)
1	100	15	1.47	0.7
2	200	21	1.84	1.2
3	300	27	2.84	1.57
4	400	38	4.89	2.57
5	500	31	8.57	4.08
6	600	30	13.14	5.9
7	700	33	21.31	9.6
8	800	37	31.17	12.98
9	900	40	43.46	22.80
10	1000	30	59.20	26.90
11	1100	37	79.45	37.83
12	1200	36	102.55	68.36
13	1300	34	127.87	60.84
14	1400	38	158.66	88.14
15	1500	68	194.56	102.4
16	1600	53	236.02	96.721
17	1700	47	287.34	156.15
18	1800	40	340.46	165.19
19	1900	50	398.91	198.15
20	2000	193	461.12	211.12



Graph 1: Run Time vs no. of nodes for series and parallel version for graph generated by Bloggers Data

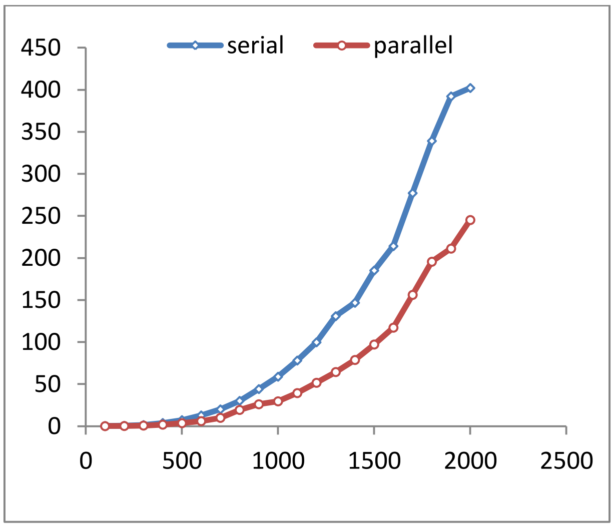
5.2 Using Erdos Renyi

The observation for Erdos Renyi graph is shown in table no. 2 and corresponding graph is shown in graph no.2. here also it can be easily observed that the algorithm is showing a speed up of about 0.5. The probability for an edge to be present was taken as 0.10 in generating the Erdos-Renyi graph.

Table 2 : Observations for Erdos Renyi

Serial no.	Number of nodes	Number of clusters	Run time (sequential)	Run time(in parallel)
1	100	21	0.18	0.16
2	200	31	0.49	0.32
3	300	33	1.42	0.81
4	400	37	3.87	1.89
5	500	39	7.29	3.64
6	600	40	12.89	6.05
7	700	39	20.09	10.00
8	800	42	30.16	19.41
9	900	43	44.19	26.15
10	1000	43	58.89	29.64
11	1100	49	78.12	39.42
12	1200	51	99.81	51.66
13	1300	50	130.85	64.45
14	1400	53	146.77	78.81
15	1500	60	185.16	97.19
16	1600	55	214.15	117.19
17	1700	70	277.15	156.19
18	1800	75	339.15	195.59
19	1900	101	392.15	211.19
20	2000	155	402.12	245.19

Graph 2: Graph for Erdos-Renyi Graph



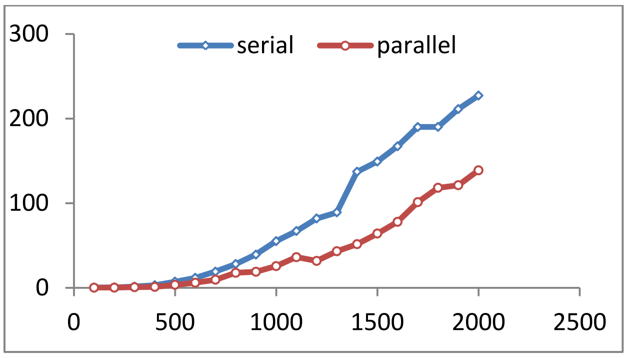
5.3 Using Barabási–Albert algorithm

The observations for Barabasi-Albert Graph are shown in Table no.3 and its corresponding graph is shown in graph 3. Here also it is evident that it is showing a speed up close to 0.50.

Table 3: Observations For Barabasi-Albert algorithm

Serial no.	Number of nodes	Number of clusters	Run time(sequential)	Run time(in parallel)
1	100	16	0.19	0.11
2	200	27	0.37	.17
3	300	36	1.39	0.71
4	400	33	3.07	1.07
5	500	31	7.31	3.51
6	600	41	11.79	5.95
7	700	37	19.09	9.51
8	800	44	28.11	17.71
9	900	46	39.18	18.91
10	1000	49	55.16	25.61
11	1100	47	67.12	36.16
12	1200	53	81.91	31.76
13	1300	51	89.15	43.19
14	1400	59	137.15	51.57
15	1500	77	149.15	64.11
16	1600	64	167.19	77.91
17	1700	81	189.91	101.2
18	1800	85	190.11	118.17
19	1900	95	211.15	121.21
20	2000	132	227.16	138.81

Graph 3: graph for observations from Barabasi-Albert Graph



6. RESULT, CONCLUSION AND FUTURE WORK

It can be easily observed from the results we are getting for different graphs i.e. the graph generated by using bloggers data also the one generated by using stochastic graph generator that MCL is showing a speed up of 0.50(approximately) and hence the analyses of large graphs can be done easily with in optimal time limits through parallelization. Most of the times community detection is performed in big social networks hence parallelization of clustering or community detection algorithm is practical approach to get computational efficiency in real world. Clustering can be extended to create recommendation systems such as blog recommendation as in our case blogs appearing in the same clusters are more relevant with the blogs that are in the same cluster than the blogs that appear in other cluster.

7. REFERENCES

[1] Sauta Elisa Schaeffer, "Survey Graph clustering," Elsevier Computer Science Review, vol. I, pp. 27-64, 2007.

[2] Venu Satuluri, "Markov Clustering of Protein Interaction Networks with Improved Balance and Scalability" ACM-BCB 2010, Niagara Falls, NY, USA.

[3] ULRIK BRANDES, MARCO GAERTLER and DOROTHEA WAGNER, "Engineering Graph Clustering: Models and Experimental Evaluation" ACM Journal of Experimental Algorithmic 12 (2007), Article 1.1.

[4] S. van Dongen. MCL - Graph Clustering by Flow Simulation. Ph.D. Thesis, University of Utrecht, 2000.

[5] P. Upadhyaya. Clustering Techniques for Graph Representations of Data. Technical report, Indian Institute of Technology Bombay, 2008.

[6] V. Satuluri and S. Parthasarathy. Scalable Graph Clustering Using Stochastic Flows: Applications to Community Discovery. KDD, 2009.

[7] Nitin Agarwal and Huan Liu and Lei Tang and Philip S. Yu}, Booktitle = {Proccedings of the First ACM International Conference on Web Search and Data Mining (Video available at: http://videolectures.net/wsdm08_agarwal_iib/)},Title = {Identifying the Influential Bloggers}, Pages = {207--218}, Url ={http://videolectures.net/wsdm08_agarwal_iib/}, Year = {2008},}

[8] Yogendra Kumar Dehariya,"Comparative Analysis of graph Clustering algorithm using Bloggers Data",ICICT 2014.