2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS)

BRDPHHC: A Balance RDF Data Partitioning Algorithm based on Hybrid Hierarchical Clustering

Yonglin Leng, Zhikui Chen, Fangming Zhong, Hua Zhong
School of Software Technology, Dalian University of Technology, 116620, Dalian, China

***Abstract* - Data partitioning is a fundamental step to achieve effective storage and query of RDF big data. This paper presents a balance RDF data partitioning algorithm based on hybrid hierarchical clustering (BRDPHHC), which combines AP and K-means clustering. BRDPHHC's functionality includes three aspects: (i) a pre-processing step combining nodes compression and nodes remove to reduce the scale of raw data points, (ii) AP clustering algorithm is used to coarsen the RDF graph step by step and produce data blocks, and (iii) K-means algorithm is used for data partitioning finally. Experiments on benchmark datasets demonstrate the effectiveness of the proposed scheme.**

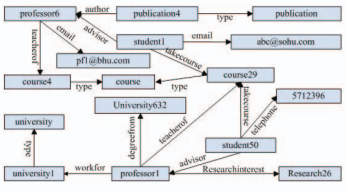
Index Terms - RDF; data partitioning; hybrid hierarchical clustering

I. INTRODUCTION

The Resource Description Framework (RDF) is the W3C's recommendation as the basement of the semantic web [1], which can represent all the identified information in the web and provide the interoperability between applications [2]. RDF data can be described as a collection of triples denoted as SPO (*subject, predicate, object*), in which the subject indicates an entity, and the object is the name of the entity or literal in predicate. It can also be represented by a directed graph, with the entities (*i.e. subjects and objects*) as nodes and the predicate as directed edges from subject to object [3]. Fig.1 (a) shows an example RDF triples, and Fig.2 (b) is the directed graph of RDF.

subject	predicate	object
Professor1	workfor	university1
Professor1	researchinterest	research26
Professor1	teacherof	course29
Student1	takecourse	course29
Student1	email	abc@sohu.com
University1	type	university
…	…	…

(a) Example of RDF triple



(b) Example of RDF directed graph
Fig. 1 RDF examples

With the wide application of RDF data, RDF data size increases sharply. The RDF data storage in single server cannot meet the requirement of massive data storage and query. To tackle this problem, the distributed storage was proposed to store the massive RDF data [3, 4]. The data partitioning is a key concern in RDF data distributed storage. In other words, a bad partitioning scheme will result in low efficiency of storage and query.

Typical data partitioning algorithms adopt horizontal or vertical partitioning based on triples [5]. This kind of algorithms ignores the correlations between triples, leading to a large number of join operations among the compute nodes during the RDF query. To address this problem, graph-based approach has been proposed to partition the RDF data, which stores the closely related nodes in the same compute node [6]. In this scheme, most of data queries can be performed in parallel to improve the query efficiency. Moreover, this scheme can reduce the join operation among the compute nodes. The representative partitioning algorithm based on graph including geometric partitioning [7], spectral partitioning [8] and multilevel partitioning algorithm [9, 10, 11].

Compared with other graph partitioning methods, the multilevel partitioning algorithm is superior to others in time complexity and the scale of data. However, this kind of algorithms mainly focuses on the partitioning efficiency, ignoring the balance of data partitioning, resulting in the difficulty in parallel to perform. In this paper, we present a balance RDF data partitioning algorithm based on hybrid hierarchical clustering (BRDPHHC), which works in three steps. Firstly, a pre-processing step is executed to reduce the number of raw data points by combining the special attributes and removing the high degree nodes. Secondly, the affinity propagation clustering algorithm (AP) is used to pre-cluster the RDF graph for producing data blocks. In each AP clustering, we merge or split the data blocks to balance the scale of the data blocks through interactive edge. Finally, an interactive edge weighted K-means clustering method is presented to partition data blocks and to realize the partition.

The rest of the paper is organized as follows. In Section 2, we present the pre-processing method for RDF graph by two steps. Section 3 depicts the similarity measurement based on adjacency and interactive edges. In Section 4, we describe the BRDPHHC algorithm for RDF data partitioning. Performance evaluation is illustrated in Section 5. Section 6 concludes the paper.

II. PRE-PROCESSING FOR RDF GRAPH

In this section, we present the proposed pre-processing steps for RDF graph. Combination of the nodes with unique

This work is supported by Project U1301253 of NSFC and Project 201202032 of Liaoning Provincial Natural Science Foundation of China.

978-1-4799-8937-9/15 $31.00 © 2015 IEEE
DOI 10.1109/HPCC-CSS-ICESS.2015.190

attribute value is first described, followed by the removing of the nodes with high degree.

A. Combination of the Nodes with Unique Attribute Values

Given an RDF graph $G=(V,E)$, $V=V_e \bigcup V_l$ represents the node set in which V_e denotes the entity node set and V_l denotes the attribute node set, while $E=\{e(v_i,v_j)\,|\,v_i,v_j \in V\}=E_r \bigcup E_a$ denotes the directed edge set in which E_r denotes the relationship edge set and E_a denotes the attribute edge set.

If the value of v_j is only belonging to the entity node v_i under the condition $(v_i,v_j)\in E_a$, then v_i and v_j must be assigned to the same storage node in the process of partitioning. For example, the node "5712396" which is the value of the attribute edge "telephone" is only belonging to the entity "student50." To improve the efficiency of partitioning, the paper combines such nodes and their corresponding entity to one node.

B. Remove of High Degree Nodes

In the real applications, some nodes have high degrees while others have low degrees. For example, over 90% nodes have only less than 5 neighbors while some nodes with more than 100, 000 neighbors in DBpedia [12]. Generally speaking, when one node has more neighbors, it is queried more frequently, leading to higher communication cost [3].

To reduce the join operations in the process of query, the paper removes the nodes with high degrees before partitioning. After partitioning, the removed nodes will be stored into such the storage nodes with the corresponding nodes.

III. SIMILARITY MEASURE

Similarity measurement is the key to the graph clustering technique. This section describes two metrics to calculate the similarity between node pairs.

A. Adjacency Measure

In this paper, we propose a metric based on adjacency. The similarity measurement based on adjacency is widely used in many graph partitioning algorithms to group nodes, which are highly connected [13]. The idea of adjacency measure can be expressed as "if the neighbors of one node u are connected with another node v , then the nodes u and v have tighter connection. In contrast, the nodes u and v will become looser." Furthermore, the shortest distance between two nodes also affect the similarity.

We use l to denote the shortest distance between two nodes. The weight between two nodes is defined as follows:

$$w_{uv}=1/l \qquad (1)$$

The similarity between two nodes is defined as follows:

$$similarity(u,v)=\frac{\sum_{k\in \mathrm{int}\, er_r(u,v)} w_{kv}}{\sum_{k\in N_r(u)} w_{ku}} \qquad (2)$$

, where $N_r(u)$ is the neighbor set within the radius r of the node u and $\mathrm{int}\, er_r(u,v)=N_r(u)\bigcap N_r(v)$ is the intersection of the neighbor sets of the node u and v .

For instance, the similarity between u and v is as shown in Fig. 2, s(u, v) =0.667 and s(u, m) =0.5. Whereas, s(a, b) =0.7143 and s(a, d)=0.5714, which have the same number of neighbors, but they have different weights, resulting in the different similarities. Here, the computation of similarity neglects the direction of edges.

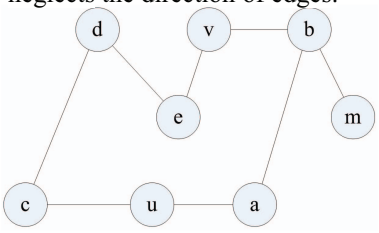


Fig. 2 An example of adjacency measure

B. Interactive Edge Measure

Interactive edge is the edge generated by the nodes in two different partition. One principle of RDF graph partitioning is to minimize the interactive edge. In hybrid hierarchical clustering, from the second layer, the data blocks in each layer are clustering collections of the previous layer. In order to minimize the interactive edges among the compute nodes, we assume that the more the number of interactive edges between two data blocks, the larger the similarity is, and the data blocks are easily clustered into a collection.

Given two data blocks C_i and C_j , $cut(C_i,C_j)$ represents the number of interactive edges. As shown in Fig.3, in the first layer, the nodes {a, b, c, d, e, f} are clustered into a data block C_1 and the nodes {g, h, i, j} are clustered into another data block C_2 , so $Cut(C_1,C_2)=5$.

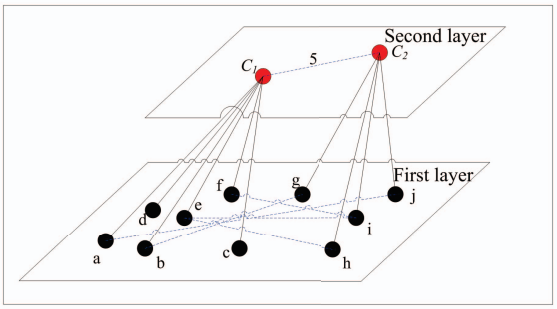


Fig. 3 Example of interactive edge

We use a linear function conversion to normalize the similarity between data blocks, where $cut_{\min}(C_k)$ and $cut_{\max}(C_k)$ denote the minimum and maximum number of interactive edges in k data blocks.

$$similarity(C_i, C_j) = \frac{cut(C_i, C_j) - cut_{\min}(C_k)}{cut_{\max}(C_k) - cut_{\min}(C_k)} \quad (3)$$

IV. BRDPHHC ALGORITHM

In this section, we first present the framework of BRDPHHC algorithm, which simultaneously considers both interactive edge and balance. Then, the detailed steps are described. The framework of BRDPHHC algorithm is shown in Fig.4.

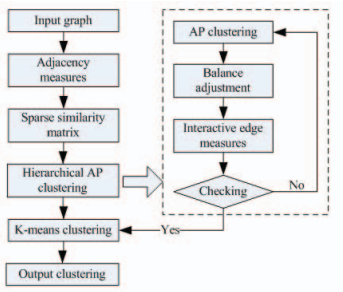


Fig. 4 Framework of BRDPHHC algorithm

A. Hierarchical AP Clustering

AP clustering algorithm was proposed by Brendan J. Frey and Delbert Dueck, which is a new clustering algorithm. Compared with the traditional clustering algorithms, AP clustering does not require a pre-specified number of clusters, and it iteratively transmits real-valued messages among the data points to gradually find the potential exemplars for forming a collection of high quality exemplars.

AP clustering algorithm takes a similarity matrix between data points as an input, where the similarity $s(i,k)$ indicates how well the data point with index k is suited to be the exemplar for data point i [13]. AP iteratively updating a responsibility matrix R, sent from data point i to candidate exemplar point k, which reflects the accumulated evidence for how well-suited point k is to serve as the exemplar for point i, considering other potential exemplars for point i, and an availability matrix A, sent from candidate exemplar point k to point i, which reflects the accumulated evidence for how appropriate it would be for point i to choose point k as its exemplar, taking into account the support from other points that point k should be an exemplar. To begin with, the availabilities matrix A are initialized to zero. Then, the responsibilities are computed using the rule.

$$r(i,k) = s(i,k) - \max_{k' \neq k} \{a(i,k') + s(i,k')\} \quad (4)$$

The following availability update gathers evidence from data points as to whether each candidate exemplar would make a good exemplar:

$$a(i,k) = \min\{0, r(k,k) + \sum_{i' \notin \{i,k\}} \max\{0, r(i',k)\}\}$$

$$a(k,k) = \sum_{i' \neq k} \max\{0, r(i',k)\} \quad (5)$$

In each iteration, there are generally n^2 data pairs whose responsibility and availability values need to be calculated, thus the computation complexity is $O(n^2)$. When the number of iterations is T, the computation complexity is $O(Tn^2)$. This greatly affects the performance especially when the data are large. It has been pointed out in [14] that the sparsity of the constructed graph will lead to faster calculation since the information propagation only needs to be performed on the existing edges. So for a sparse similarity matrix, the complexity will up to $O(Tn)$ [15].

The goal of RDF graph partitioning is to cluster the closely connected nodes in a compute node. So the smaller similarity between two nodes, the less possibility two nodes cluster together. In order to improve the efficiency of clustering, we set the similarity equal to $-\infty$ if the similarity values less than δ. As a result, the sparsity of the matrix improved and time complexity in AP clustering is reduced.

Algorithm 1: Hierarchical AP Clustering Algorithm(HC)

Input: RDF graph $G = (V, E)$, cluster threshold T

Output: Cluster set $C = \{C_1, C_2, \cdots, C_m\}$,where $m \leq T$

Method:

1. Construct a sparse similarity S based on (2)
2. Execute AP clustering on matrix S and generate m clusters
3. If $m > T$, compute the similarity among the m clusters using (3) to produce a new similarity matrix S
4. Take S as a new input，continue to run 2, until $m \leq T$

B. Partitioning-balance Adjustment

The balance of the subgraph scales plays an important role in distributed storage and query when a huge graph is partitioned into several subgraphs. Specially, a bad balance will lead to a low efficiency of query. Therefore, to preserve the balance, the partitioning-balance will be adjusted in each clustering.

Given a graph $G = (V, E)$, we divide the graph into k partitions $P = \{P_1, P_2, \cdots, P_k\}$. The partitioning-balance of a k-way partitions should satisfy $1 - e_1 \leq PB_i \leq 1 + e_2$, where $PB_i = |V_i| / m$ and $m = |V| / k$. The partitioning-balance adjustment approach is outlined in Algorithm 2.

Algorithm 2: Partitioning-balance Adjustment

Input: Partition $P = \{P_1, P_2, \cdots, P_k\}$

Output: New partitioning-balance $P' = \{P_1', P_2', \cdots, P_t'\}$

Method:

1. Compute partitioning-balance $PB = \{PB_1, PB_2, \cdots, PB_k\}$.
2. For p =1 to k

if $PB_i < 1 - e_1$

$merge(P_i, P_j)$, where $Cut(P_i, P_j)$ is max

and $PB_j < 1 + e_2$.

else if $PB_i > 1 + e_2$

Partition P_i use KL algorithm

until $1 - e_1 \le P'_i \le 1 + e_2$.

C. Final Partitioning

During the Hierarchical AP clustering, we use the AP clustering to gradually reduce the scale of the RDF graph. But the number of clusters in AP clustering is uncertain because of the influence of preferences. When the compute node is specified, AP clustering cannot achieve the final partition. In this phase, we use K-means clustering method to perform the final clustering.

Algorithm 3: K-means Clustering

Input: interactive edge matrix S, the number of clusters k

Output: cluster set $C = \{C_1, C_2, \cdots, C_k\}$

Method:

1. Select k initial cluster centers $C = \{c_1, c_2, \ldots, c_k\}$.
2. Assign each data object to its closest center according to similarity between data objects and cluster centers.
3. Update the cluster centers.

1) Compute the average vector $S(\overline{v_i})$ of cluster C_i.

$$S(\overline{v_i}) = \frac{1}{|C_i|} \sum_{v_k \in C_i} S(v_k, v_j), \forall v_j \in V$$

2) Find the new cluster center c'_i.

$$c'_i = \arg\min_{v_k \in C_i} \left\| S(v_k) - S(\overline{v_i}) \right\|$$

4. Repeat step 2 and 3, until the clustering objective E function converges.

$$E = \sum_{i=1}^{k} \sum_{v_k \in C_i} \left\| S(v_k) - S(\overline{v_i}) \right\|^2$$

D. Complexity Analysis

There are three main operations in the designed algorithm. The first one is the similarity calculation with the time complexity of $O(n^2)$. The second one is the Hierarchical AP clustering with the time complexity of $O(Tn)$, where T is the number of iterations. The last one is the k-means clustering with the time complexity of $O(KTm)$, where K is the number of clusters and m is the number of clustering collections. Therefore, the total time complexity of BRDPHHC is approximately $O(n^2)$.

V. Experiments

A. Datasets and Environment

In this section, we present a set of experiments on the Lehigh University Benchmark (LUBM) [16] and the DBLP Bibliography datasets [17]. The number of edges and nodes are shown in Table I. All experiments were performed on an i3 3.30GHz PC running Windows XP with 4GB main memory.

TABLE I STATISTICS OF DATASETS USED IN EXPERIMENTS

	|V|	|E|	Degree		First merge
			min	max	
LUBM	7171	14901	1	1419	4085
DBLP	5035	17277	1	256	3682

B. Interactive Edge Ratio and the Time Performance

Given a graph $G = (V, E)$, we divide the graph into k clusters $P_1 = (V_1, E_1), P_2 = (V_2, E_2), \ldots, P_k = (V_k, E_k)$, then the interactive edge ratio can be described as follows:

$$IER = \frac{\sum_{i=1}^{k} |cut(P_i, G \setminus P_i)|}{|E|}, \quad (6)$$

, where $cut(P_i, G \setminus P_i)$ is the number of edges between P_i and the rest of the graph. The adjacency measurement radius r is set to 2 in the experiment. Table II shows the number of removed nodes and edges under r=2 of different datasets. Fig. 5 shows the interactive edge ratio on different dataset and Fig. 6 presents runtime of the partition.

TABLE II STATISTICS OF REMOVE NODES AND EDGES

		Remove condition	Remove nodes	Remove edges
LUBM	0	—	0	0
	1	>500	3	2081
	2	>100	8	3277
	3	>60	13	3602
	4	>40	18	3841
DBLP	0	—	0	0
	1	>500	0	0
	2	>100	16	3070
	3	>60	25	3780
	4	>40	41	4582

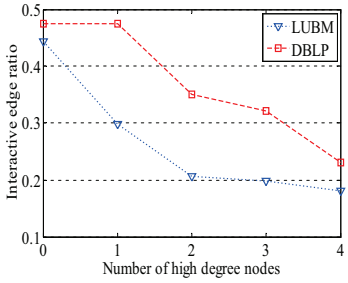


Fig. 5 Interactive edge ratio on different remove condition

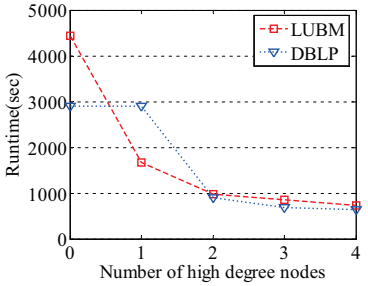


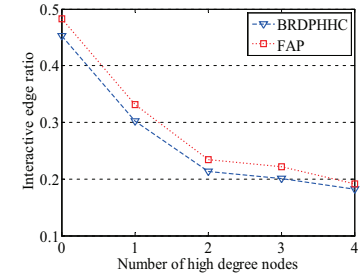
Fig. 6 Partition efficiency on different remove condition

From the result, we can see that the interactive edge ratio decreases and the time performance improves with the remove of high degree nodes.

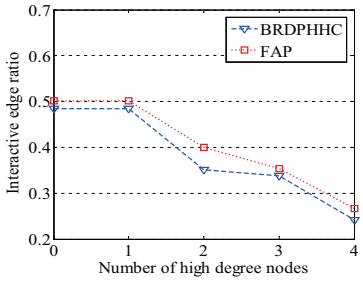
FAP is a multilevel graph partitioning method, which uses a Fast Sampling algorithm to coarsen the input sparse graph and chooses a small number of final representative exemplars. For these exemplars, FAP uses a density-weighted spectral clustering to partition the exemplars. Finally, all data points can be assigned through the corresponding representative exemplars [10].

In this paper we compared BRDPHHC algorithm and FAP algorithm in interactive edge ratio and runtime. The results are shown in Fig.7 and Fig.8.

Fig.7 (a) and (b) present the interactive edge ratio in LUBM and DBLP dataset respectively. It can be seen that the interactive edge ratio of FAP algorithm is higher than BRDPHHC.

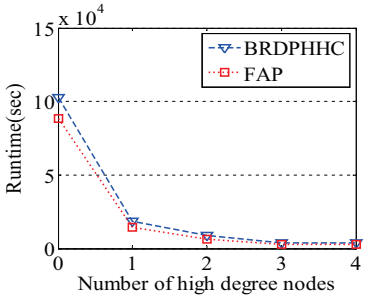


(a) LUBM

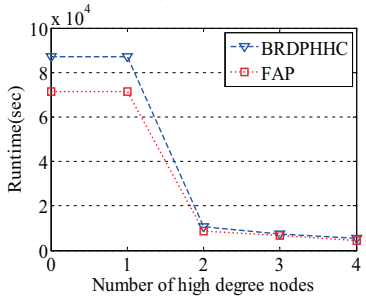


(b) DBLP

Fig. 7 Edge-cut on different datasets



(a) LUBM



(b) DBLP

Fig. 8 Partition efficiency on different datasets

C. *Partitioning-Balance*

Table III and Table IV shown the comparison of partitioning-balance between BRDPHHC and FAP in LUBM and DBLP datasets.

We calculate $PB_{\max}$ and $PB_{\min}$ using (10) and (11).

$$PB_{\max} = \max(|V_i|) / m \quad (10)$$

$$PB_{\min} = \min(|V_i|) / m \quad (11)$$

From Table III and Table IV, the balance obtained by the proposed algorithm is better than that obtained by FAP. Furthermore, the removing of the nodes with high degree results in a good balance, which demonstrates the effectiveness of the proposed algorithm.

TABLE III PARTITIONING BALANCE OF LUBM

Remove condition	BRDPHHC		FAP	
	PB_{max}	PB_{min}	PB_{max}	PB_{min}
—	1.034	0.942	1.210	0.742
>500	1.033	0.945	1.169	0.819
>100	1.017	0.980	1.134	0.836
>60	1.016	0.980	1.096	0.897
>40	1.013	0.977	1.087	0.914

TABLE IV PARTITIONING-BALANCE OF DBLP

Remove condition	BRDPHHC		FAP	
	PB_{max}	PB_{min}	PB_{max}	PB_{min}
—	1.053	0.959	1.134	0.836
>500	1.053	0.959	1.134	0.836
>100	1.019	0.974	1.095	0.907
>60	1.017	0.978	1.064	0.936
>40	1.045	0.933	1.057	0.957

VI. CONCLUSION

In this paper, we proposed a balance RDF data partitioning algorithm based on hybrid hierarchical clustering, including AP clustering and K-means algorithm. Firstly, for

the sake of parallelization, balance adjustment algorithm is used to ensure the balance in each level AP clustering. Further, to improve the efficiency of RDF graph partitioning, high degree nodes are removed before clustering. Finally, the experimental results demonstrate that our proposed algorithm outperforms FAP by a better balance.

REFERENCES

[1] RDF model and syntax specification.1999.http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

[2] F. Du, Y. Chen, X. Du, et al, "Survey of RDF query processing techniques," Journal of Software, vol. 24, no. 6, pp: 1222-1241, 2013.

[3] K. Zeng, J. Yang, H. Wang, et al, "A distributed graph engine for web scale RDF data," Proceedings of the VLDB Endowment, vol. 6, no. 4, pp: 265-276, 2013.

[4] Q. Zhang, Z. Chen, "A weighted kernel possibilistic c-means algorithm based on cloud computing for clustering big data," International Journal of Communication Systems, vol. 27, no. 9, pp: 1378-1391, 2014.

[5] K. Lee, L. Liu, Y. Tang, et al, "Efficient and customizable data partitioning framework for distributed big RDF data processing in the cloud," 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD), pp: 327-334, 2013.

[6] J. Huang, D. Abadi, K. Ren, "Scalable SPARQL querying of large RDF graph," Proc. of the VLDB endowment, vol. 4, no. 11, pp: 1123-1134, 2011.

[7] M. Heath, P. Raghavan, "A Cartesian parallel nested dissection algorithm," SIAM Journal on Matrix Analysis and Applications, vol. 16, no. 1, pp: 235-253, 1995.

[8] A. Pothen, H. Simon, K. Liou, "Partitioning sparse matrices with eigenvectors of graphs," SIAM Journal on Matrix Analysis and Applications, vol. 11, no. 3, pp: 430-452, 1990.

[9] METIS, http://glaros.dtc.umn.edu/gkhome/views/metis.

[10] F. Shang, L. Jiao, J. Shi, et al, "Fast affinity propagation clustering: A multilevel approach," Pattern recognition, vol. 45, no. 1, pp: 474-486, 2012.

[11] R. Ma, X. Wang, J. Ding, "Multilevel core-sets based aggregation clustering algorithm," Journal of Software, vol. 24, no. 3, pp: 490-506, 2013.

[12] S. Auer, C. Bizer, G. Kobilarov, et al, "DBpedia: A nucleus for a web of open data," Springer Berlin Heidelberg, 2007.

[13] S. Schaeffer, "Graph clustering," Computer Science Review, vol. 1, no. 1, pp: 27-64, 2007.

[14] B. Frey, D. Dueck. "Clustering by passing messages between data points," Science, vol. 315, no. 5814, pp: 972-976, 2007.

[15] Y. Jia, J. Wang, C. Zhang, et al, "Finding image exemplars using fast sparse affinity propagation," Proceedings of the 16th ACM international conference on Multimedia, ACM, pp: 639-642, 2008.

[16] Y. Guo, Z. Pan, J. Heflin, "LUBM: A benchmark for OWL knowledge base systems," Web Semantics: Science, Services and Agents on the World Wide Web, vol. 3, no. 2, pp: 158-182, 2005.

[17] J. Tang, J. Sun, C. Wang, et al, "Social influence analysis in large-scale networks," Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining ACM, pp: 807-816, 2009.